НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 379.8

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Задача контентної фільтрації для організації конференцій*»

Виконав: студент *4* курсу, групи *ІС-52*
Брайко Костянтин Анатолійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Ковтунець О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *старший викладач Халус О.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доцент каф. ОТ, к.т.н. Верба О.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Брайко К. А. _____
(підпис)

Київ – 2019 р.

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 12 рисунків, 44 таблиць, 1 додаток, 6 джерел.

Дипломний проект присвячений розробці задачі контентної фільтрації для організації конференцій.

Призначенням розробки є вибір місця для проведення конференції за рахунок алгоритму колаборативної фільтрації.

Метою розробки є спрощення процесу пошуку та вибору місця для проведення конференції.

У розділі інформаційного забезпечення здійснюється опис вхідних та вихідних даних, демонструється схема структурна бази даних з детальним описом кожної таблиці.

Розділ математичного забезпечення присвячений знаходженню та обґрунтуванню методу вирішення сформульованої постановки задачі.

Програмне забезпечення описує вибір засобів розробки та спроектовану архітектуру програмного забезпечення.

У технологічному розділі описується керівництво користувача та випробування програмного продукту.

Ключові слова: КОНТЕНТНА ФІЛЬТРАЦІЯ, АЛГОРИТМ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ, БАЗА ДАНИХ.

ABSTRACT

Structure and scope of work. Explanatory note of the diploma project consists of five sections, contains 12 drawings, 44 tables, 1 supplement, 6 sources.

The diploma project is devoted to the development of the task of content filtering for the organization of conferences.

The purpose of the development is the choice of place for the organization of the conference due to the collaborative filtering algorithm.

The goal of the development is to simplify the process of finding and choosing a place to host the conference.

In the information support section, a description of the input and output data is provided, a structured database schema with a detailed description of each table is displayed.

The section of the mathematical support is devoted to finding and substantiating the method of solving the formulation of the problem.

The software describes the choice of development tools and the design of the software architecture.

The technology section describes the user guide and software test.

Key words: CONTENT FILTRATION, ALGORITHM OF COLLABORATIVE FILTRATION, DATABASE.

ЗМІСТ

ВСТУП

Дипломний проект присвячений розробці задачі контентної фільтрації для організації конференцій. Причиною появи такої ідеї слугував сам процес пошуку місця для проведення конференції.

Організація будь-якої конференції починається спочатку із вибору місця для її проведення. Зазвичай, щоб знайти потрібне місце організатори переглядають десятки оголошень розміщених в мережі Інтернет, крім того вони далеко не завжди містять всю потрібну інформацію або ключова інформація не структурована, що впливає на складність оцінки відповідного оголошення. Але навіть у випадку, якщо оголошення задовольняє організатора, він повинен телефонувати особі, яка розмістила оголошення, для уточнення можливості оренди приміщення на визначену дату та час. Весь описаний процес доволі складний та тривалий, тому виникла потреба в його автоматизації.

Практичне значення одержаних результатів. Реалізовано алгоритм колаборативної фільтрації.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Щомісячно в Україні проводяться сотні конференцій, успіх яких залежить не лише від досвідченості доповідачів та піднятих тем, але й від обраного місця проведення.

Процес вибору місця зазвичай починається з пошуку оголошень. Це можуть бути як розклеєні на вулиці, так і опубліковані в мережі Інтернет відомості про місця, придатні для проведення конференцій. Якщо вуличні оголошення надають мінімум інформації, то електронні – є більш інформативними.

Більшість оголошень містить інформацію про локацію, місткість приміщення, погодинну орендну плату, ступінь технічного забезпечення(наявність проектора, мікрофона, підсилювача звуку, фліпчарта, холодильника, кухні та інше), розклад роботи, правила, контактні дані орендодавця та інше. Якщо конкретне оголошення зацікавило клієнта, щоб дізнатися про доступність локації на конкретний день та забронювати її, він повинен зателефонувати орендодавцю. Причому це не гарантує досягнення домовленості, оскільки більшість конференцій проводяться в однакові періоди, що ускладнює пошук вільної локації.

Таким чином, постає необхідність розробки системи, яка могла би пришвидшити пошук місця для проведення конференції. Дана система повинна пришвидшити як сам пошук потрібного оголошень, за рахунок фільтрації по ключовим даним(місткість, погодинна орендна плата, місто, додаткові умови) та порівняння оголошень, так і сам процес ідентифікації незайнятості та бронювання локації.

1.1.1 Опис процесу діяльності

Об’єктом автоматизації є процес вибору місця для проведення конференції.

Розглянемо дії, які має виконати користувач для вибору місця проведення конференції, за допомогою схеми структурної діяльності. Схема структурна діяльності наведена у графічному матеріалі.

1.1.2 Опис функціональної моделі

Виділимо акторів системи та опишемо дії, які може виконувати кожен актор у системі. Акторами системи є орендодавець, користувач та клієнт.

Нижче наведений опис кожного з акторів.

Орендодавець. Особа зацікавлена в здачі нерухомого майна в оренду для проведення конференцій. Система надає можливість створювати оголошення в спеціалізованому конструкторі та зберігати їх в базі даних.

Користувач. Неавторизований користувач сайту. Має права лише на фільтрацію, порівняння та перегляд оголошень.

Клієнт. Авторизований користувач сайту, крім всіх прав, які має користувач, має доступ до контактної інформації, може переглядати рекомендовані оголошення та має можливість залишати коментар та виставляти оцінку по кожній пропозиції.

Дії, які можуть виконувати актори системи представлені в схемі структурній варіантів використання, яка наведена в графічному матеріалі.

1.2 Огляд наявних аналогів

В результаті пошуку аналогів була виявлена система зі схожими функціями: **Renty**.

Основними особливостями системи **Renty** є:

- використання фільтрів по місту, типу події, кількості гостей, ціні та додаткових умовах(наявність додаткового обладнання, кухні та інше);
- впорядкування пропозицій по популярності, ціні, місткості;
- використання Google Maps для демонстрації положення локацій на карті;
- можливість порівняння кількох пропозицій;
- можливість оцінити локацію та залишити коментар.

Недоліками вище наведеної системи, на мою думку, є:

- доступні лише чотири міста: Київ, Одеса, Харків, Дніпро;
- дана система має широку спеціалізацію, вона використовується для пошуку локацій як для проведення ділових заходів(тренінг, переговори, конференція, лекція та ін.), так і для розважальних(день народження, банкет, весілля).
- широка спеціалізація є причиною появи надлишкових фільтрів, які є спільними для всіх можливих заходів(можна вважати як плюсом так і мінусом). Тобто може існувати локація, створена випадково або зумисно, для проведення конференції в якій передбачений танцпол.

Використання колаборативного алгоритму буде характерною відмінністю моєї розробки від розглянутої вище. Хоча сервіс **Renty** також маю подібну функціональність, але рекомендаціями там виступають тільки оголошення з максимально подібними характеристиками відносно переглянутої користувачем, а виставлені оцінки слугують лише для визначення рейтингу локації. Натомість суть моєї розробки полягає у визначенні клієнтів з максимально подібними смаками, аналізуючи їхні оцінки, та прогнозуванні оцінок для невідомих клієнту оголошень, з метою побудови рекомендацій. Крім того в системі передбачається функціональність для бронювання залу.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням розробки є вибір місця для організації конференції за рахунок колаборативної фільтрації.

1.3.2 Цілі та задачі розробки

Метою розробки є спрощення процесу пошуку та вибору місця для проведення конференції.

Для досягнення поставленої мети необхідно реалізувати наступні задачі:

— можливість перегляду оголошень;
— можливість залишити коментар та виставити оцінку;
— фільтрація оголошень;
— порівняння оголошень;
— можливість бронювання залу;
— формування рекомендацій оголошень;
— можливість створення оголошень орендодавцем;
— вхід в систему двох типів користувачів;
— забезпечення зв’яку між клієнтом та орендодавцем.

Висновок до розділу

У даному розділі описано предметне середовище; наведені структурні схеми діаграми діяльності та варіантів використання з описом кожного актора та виконуваних ним функцій в системі; здійснено огляд наявних аналогів з визначенням їх характерних ознак та недоліків; наведено призначення, цілі та задачі розроблювальної системи.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані задачі контентної фільтрації для організації конференцій надходять з декількох джерел, а саме від:

- орендодавці;
- клієнти(користувачі).

Дані вводяться в систему через інтерфейс десктопного застосування. Розглянемо структуру даних, які надходять від орендодавців та клієнтів.

Дані, які надходять від орендодавців:

- реєстрація:
    1) ім’я;
    2) мобільний телефон;
    3) електронна пошта;
    4) логін;
    5) пароль.
- створення оголошення:
    1) зображення локації;
    2) назва локації;
    3) адреса локації;
    4) опис локації;
    5) місткість локації;
    6) сума погодинної оренди;
    7) наявність проектора;
    8) наявність мікрофона;
    9) наявність телевізора;
    10) наявність дошки;
    11) наявність холодильника;
    12) наявність кухні;

13) правила орендодавця;

14) розклад роботи;

Дані, які надходять від клієнтів:

- реєстрація:
  1) ім’я;
  2) мобільний телефон;
  3) електронна пошта;
  4) логін;
  5) пароль.
- пошук оголошення:
  1) місткість локації;
  2) сума погодинної оренди;
  3) місто;
  4) наявність проектора;
  5) наявність мікрофона;
  6) наявність телевізора;
  7) наявність дошки;
  8) наявність холодильника;
  9) наявність кухні;
  10) наявність підсилювачів звуку.
- оцінювання локації:
  1) оцінка;
  2) коментар.
- резервування локації:
  1) дата;
  2) інтервал часу;
  3) контактні дані.

2.2 Вихідні дані

Вихідні дані, які формуються на основі заповненої форми пошуку оголошень:

- список оголошень, які задовольняють введеним критеріям пошуку.

Вихідні дані, які формуються при виставленні оцінок:

- список рекомендованих оголошень, які можуть зацікавити клієнта при майбутньому пошуку.

Вихідні дані, які формуються при резервуванні локації:

- назва локації;
- зарезервований день;
- зарезервований інтервал часу;
- ім’я користувача;
- контактні дані користувача.

2.3 Опис структури бази даних

При проектуванні бази даних було виділено шість сутностей, які описані в таблицях 2.1 – 2.6.

Таблиця 2.1 – Таблиця клієнтів

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Client	id_client	integer	Код, під яким клієнт зареєстрований в базі даних
	name	varchar(30)	Ім’я клієнта
	mobile	varchar(13)	Номер мобільного телефону
	email	varchar(20)	Електронна пошта
	login	varchar(40)	Логін
	password	varchar(40)	Пароль

Таблиця 2.2 – Таблиця орендодавців

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Homeowner	id_homeowner	integer	Код, під яким орендодавець зареєстрований в базі даних
	name	varchar(30)	Ім’я орендодавця
	mobile	varchar(13)	Номер мобільного телефону
	email	varchar(20)	Електронна пошта
	login	varchar(40)	Логін
	password	varchar(40)	Пароль

Таблиця 2.3 – Таблиця оголошень

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Advertisement	id_advertisement	integer	Код, під яким оголошення зареєстроване в базі даних
	id_homeowner	integer	Посилання на орендодавця
	image	image	Фотографія локації
	name	varchar(50)	Назва локації
	address	varchar(30)	Адреса залу
	roominess	integer	Місткість залу
	price	float	Ціна погодинної оренди
	city	varchar(20)	Місто, в якому знаходиться зал
	TV	boolean	Наявність телевізора в залі
	speakers	boolean	Наявність колонок в залі
	microphone	boolean	Наявність мікрофона в залі
	projector	boolean	Наявність проектора в залі
	flip_chart	boolean	Наявність дошки в залі
	refrigerator	boolean	Наявність холодильника на локації
	kitchen	boolean	Наявність кухні на локації
	description	text	Опис оголошення
	rules	text	Правила від орендодавця
	shedule	text	Розклад роботи залу

Таблиця 2.4 – Таблиця оцінок

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Rating	id_rating	integer	Код, під яким оцінка записана в базі даних
	id_advertisement	integer	Посилання на оголошення
	id_client	integer	Посилання на клієнта
	mark	integer	Оцінка
	comment	text	Коментар

Таблиця 2.5 – Таблиця рекомендацій

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Recommendation	id_recommendation	integer	Код, під яким рекомендація записана в базі даних
	id_advertisement	integer	Посилання на оголошення
	id_client	integer	Посилання на клієнта

Таблиця 2.6 – Таблиця резервувань

Назва таблиці	Назва стовпця	Тип даних	Детальна інформація
Reservation	id_reservation	integer	Код, під яким оцінка записана в базі даних
	id_client	integer	Посилання на клієнта
	id_advertisement	integer	Посилання на оголошення
	date	date	Зарезервована дата
	time	varchar(11)	Зарезервований інтервал часу

Схема структурна бази даних в якій представлено всі вище описані сутності представлена в графічному матеріалі.

Висновок до розділу

У даному розділі описані вхідні дані, які система приймає на вхід від орендодавців та користувачів, та вихідні дані, які отримуються після обробки вхідних даних. У випадку орендодавця вхідні дані надходять у таких ситуаціях: при реєстрації та створенні оголошення; у клієнта вхідні дані надходять при реєстрації, пошуку оголошень, оцінюванні локації та резервуванні локації.

Крім того у даному розділі створено фізичну модель бази даних з описом всіх таблиць та полів.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Сервіс з пошуку залу для проведення конференції у своїй функціональності містить можливість надання рекомендацій зареєстрованим клієнтам. Загалом в базі даних системи збережено множину клієнтів, кількістю **n,** та активних оголошень, кількістю **m**.

У системі реалізована можливість оцінювання оголошень клієнтами, спираючись на власне суб'єктивне враження від наданих орендодавцем послуг. Клієнт за власним бажанням може виставити оцінку оголошенню. Максимально можлива оцінка, передбачена для виставлення, дорівнює **b**.

Таким чином, проблема полягає в пошуку оголошень, які можуть зацікавити конкретно взятого клієнта. Дані оголошення будуть представлені у вигляді рекомендацій. Кількість рекомендованих оголошень повинна бути не більша **семи**(клієнт повинен мати час переглянути їх всі).

3.2 Математична постановка задачі

Дано:

- кількість зареєстрованих клієнтів **n**;
- кількість опублікованих оголошень **m**;
- множина зареєстрованих клієнтів $\mathbf{U=\{u_1, u_2, \dots, u_n\}}$;
- множина опублікованих оголошень $\mathbf{V=\{v_1, v_2, \dots, v_n\}}$;
- матриця оцінок **R**, де $\mathbf{R_{u_i v_j}}$ - оцінка виставлена **j**-тому оголошенню **i**-тим клієнтом;
- клієнт $\mathbf{u_k}$, якому потрібно видати рекомендації;
- максимально можлива оцінка для виставлення **b**.

Змінні:

- $\boldsymbol{x_j} = \begin{cases} 1 \\ 0 \end{cases}$, оголошення $\mathbf{v_j}$ рекомендоване або не рекомендоване;
- $\boldsymbol{S_{ui}}$ – числова характеристика подібності **i**-того клієнта до **k**-ого;
- $\boldsymbol{R'_{ukvj}}$ – прогнозована оцінка **j**-ого оголошення **k**-тим клієнтом.

Обмеження:

- обмеження на кількість рекомендованих оголошень:

$$\sum_{j=1}^{m} x_j \leq 7 \tag{3.1}$$

- подібність клієнтів через формулу косинусної міри:

$$S_{ui} = \frac{\sum_{j \in V_{uiuk}} R_{uivj} \times R_{ukvj}}{\sqrt{\sum_{j \in V_{ui}} R^2{}_{uivj}} \times \sqrt{\sum_{j \in V_{uk}} R^2{}_{ukvj}}} \tag{3.2}$$

- прогнозована оцінка:

$$R'_{ukvj} = \frac{\sum_{i=1}^{n} S_{ui} \times R_{uivj}}{\sum_{i=1}^{n} |S_{ui}|} \tag{3.3}$$

- обмеження на прогнозовану оцінку:

$$R'_{ukvj} \geq \frac{b}{2} \tag{3.4}$$

Цільова функція:

$$Z = \left(\sum_{j=1}^{m} (x_j \times R'_{ukvj})\right) \rightarrow max \tag{3.5}$$

3.3 Обґрунтування методу розв’язання

Відповідно до змістовної постановки завдання описана систему можна класифікувати як рекомендаційну. Зазвичай у рекомендаційних системах використовується один з двох базових підходів: колаборативна фільтрація (collaborative filtering) та контентна фільтрація (content-based filtering). Також існує клас підходів, що базуються на поєднанні двох основних–гібридна фільтрація (hybrid filtering).

Алгоритм контентної фільтрації є незалежним від інших користувачів, рекомендації формуються на основі конкретного користувача, та не враховується інформація інших клієнтів системи. Натомість алгоритм колаборативної фільтрації використовує відомі переваги(оцінки) групи користувачів для прогнозування невідомих переваг(оцінок) іншого користувача. За допомогою цього алгоритму будується певна таблиця користувачів, які групуються за схожістю, та прогнозуються результати для інших користувачів [1].

В нашому випадку доцільно використати саме алгоритм колаборативної фільтрації, оскільки саме цей алгоритм дозволить збільшити їх варіативність, а не одноплановість.

3.4 Опис методів розв’язання

Основною математичною задачею, яку розв’язує даний алгоритм, є задача прогнозування оцінки невідомого для клієнта оголошення.

Нехай нам відома таблиця оцінок по кожному клієнту та оголошенню:

	V1	**V2**	**…**	**Vm**
U1	**R11**	**R12**	**…**	**R1m**
U2	**R21**	**R22**		**R2m**
…	**…**			
Un	**Rn1**	**Rn2**	**…**	**Rnm**

Нехай нам потрібно спрогнозувати оцінку клієнта Uk, k<n, по всім не оцінених оголошеннях.

Знайдемо числову характеристику подібності кожного клієнта до Uk, використовуючи формулу (3.2):

U1	**U2**	**…**	**Uk**	**…**	**Un**
S1	**S2**	**…**	**Sk**	**…**	**Sn**

Причому для кожного S виконується: $0 \leq S_i \leq 1,\ i = \overline{1,n}$.

Помножимо матрицю оцінок на відповідні значення S. Отримаємо матрицю наступного вигляду:

	V1	**V2**	**…**	**Vm**
U1	**S1*R11**	**S1*R12**	**…**	**S1*R1m**
U2	**S2*R21**	**S2*R22**		**S2*R2m**
…	**…**			
Un	**Sn*Rn1**	**Sn*Rn2**	**…**	**Sn*Rnm**

Тепер за формулою (3.3), знайдемо прогнозовані оцінки для клієнта Uk:

$$\mathbf{R'_{UkV1}} = \frac{(\mathbf{S1} * \mathbf{R11}) + (\mathbf{S1} * \mathbf{R21}) + \cdots + (\mathbf{Sn} * \mathbf{Rn1})}{|\mathbf{S1} + \mathbf{S2} + \cdots + \mathbf{Sn}|};$$

$$\mathbf{R'_{UkV2}} = \frac{(\mathbf{S1} * \mathbf{R12}) + (\mathbf{S1} * \mathbf{R22}) + \cdots + (\mathbf{Sn} * \mathbf{Rn2})}{|\mathbf{S1} + \mathbf{S2} + \cdots + \mathbf{Sn}|};$$

… …

$$\mathbf{R'_{UkVn}} = \frac{(\mathbf{S1} * \mathbf{R1m}) + (\mathbf{S1} * \mathbf{R2m}) + \cdots + (\mathbf{Sn} * \mathbf{Rnm})}{|\mathbf{S1} + \mathbf{S2} + \cdots + \mathbf{Sn}|}.$$

Впорядкуємо отримані оцінки за незростанням. Виберемо почергово ті оцінки при яких виконуються обмеження (3.1) та (3.4), приймем відповідне значення змінної $x_j = 1$. Відповідні оголошення і будуть рекомендаціями.

Узагальнимо описаний вище метод у вигляді покрокового алгоритму:

Крок 1. ЗНАЙТИ коефіцієнт подібності для кожного клієнта відносно шуканого.

Крок 2. ЗНАЙТИ прогнозовані оцінки для неоцінених клієнтом оголошень.

Крок 3. ВИКОНАТИ операцію сортування за незростанням для прогнозованих оцінок.

Крок 4. ВИБРАТИ першу оцінку із відсортованих.

Крок 5. ЯКЩО виконуються обмеження (3.1) та (3.4) прийняти відповідну змінну x рівною одиниці. ІНАКШЕ СТОП.

Крок 6. ВИБРАТИ наступну оцінку із відсортованих та ПЕРЕЙТИ на Крок 5.

Розглянемо роботу алгоритму на конкретному прикладі:

Нехай маємо наступну матрицю оцінок:

	V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**
U1	9	3			4				
U2	7					4		6	5
U3	6	10	5	9	8	5			7
U4	3		4	7		7	1	5	

Потрібно визначити, які оголошення порекомендувати клієнту U2.

Знайдемо коефіцієнти подібності для кожного клієнта до U2:

S1 = 0,545;

S3 = 0,443;

S4 = 0,577.

Знайдемо прогнозовані оцінки для неоцінених клієнтом оголошень.

R2 = 3,878;

R3 = 2,89;

R4 = 5,128;

R5 = 3,659;

R7 = 0,368.

Відсортуємо за незростанням отримані оцінки.

$$R4 \geq R2 \geq R5 \geq R3 \geq R7.$$

Виберемо першу оцінку із відсортованих R4. Для неї виконується умова (3.1) та (3.4). Тому $x_4 = 1$.

Виберемо наступну оцінку із відсортованих R2. Для неї не виконується умова (3.4).

Отже, оголошення V4 можна порекомендувати клієнту U2.

Висновок до розділу

У даному розділі описана змістовна постановка задачі та побудована її математична постановка, яка включає змінні, обмеження та цільову функцію. Крім того в даному розділі визначені можливі рішення поставленої задачі, вибрано та обгрунтовано метод, який реалізовано в даному дипломному проекті, з приведенням покрокового алгоритму та прикладу розв’язання задачі на тестових даних.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

При створенні програмного продукту була застосована технологія WPF з використанням мови програмування C#, мови розмітки XAML, ORM Entity Framework та системи управління базами даних Microsoft SQL Server.

WPF(Windows Presentation Foundation) - платформа, яка дозволяє створювати клієнтські застосування для систем сімейства Windows. В WPF міститься незалежний векторний модуль візуалізації, який використовує можливості сучасного графічного обладнання. Можливості векторного модуля є розширювані через можливості мови розмітки XAML. Дана платформа входить в .NET Framework, тому для розробника доступні інші бібліотеки класів .NET Framework [2].

Основні переваги платформи WPF :

- можливість використовувати мови програмування C#, VB, C/C++ та ін. для написання застосувань;
- апаратне прискорення графіки;
- розширюваний набір елементів управління і бібліотек класів;
- поділ візуальної частини та бізнес-логіки;
- можливість обробки подій;
- можливість прив’язки даних;
- розмітка, визначаюча зовнішній вигляд, тісно не пов’язана з кодом, визначаючим поведінку.

XAML – мова розмітки, основана на XML, яка відповідає за створення візуального вигляду застосування в декларативній формі. Зазвичай використовується для створення вікон, сторінок та користувацьких елементів управління, а також їх наповнення графічними елементами та елементами управління [3].

Основні особливості мови розмітки XAML :

- базується на XML;
- можливість виконувати код без компіляції.

Entity Framework – надає спеціальну об’єктно – орієнтовану технологію для роботи з даними на базі .NET Framework. Дана технологія являє собою проміжну ланку між базою даних та самим застосуванням, що дозволяє працювати з таблицями, індексами, ключами не на фізичному рівні, а на концептуальному, оперуючи об’єктами(таблиці) та їхніми атрибутами(ключі) [4].

Основні особливості технології Entity Framework:

- надає вищий рівень абстракції порівняно з іншими ORM;
- незалежність від вибору бази даних;
- використання спеціальної мови запитів LINQ;
- викоритання міграцій даних.

C# — об’єктно-орієнтована мова програмування для платформи .NET. Розроблена в 2000 році Андерсом Хейлсбергом, Скоттом Вилтамутом і Пітером Гольде під егідою Microsoft Research. Мова заснована на строгій компонентній архітектурі і реалізує передові механізми забезпечення безпеки коду(«збирач сміття», обробка винятків, безпека типів та ін.). На C # повністю написана і сама технологія Entity Framework [5].

Ключові особливості мови C#:

- використання єдиної бібліотеки класів – CLR;
- код зібраний воєдино (декларації і реалізації об'єднані разом);
- компонентна орієнтованість;
- уніфікована система типів і їх безпечність;
- автоматична і мануальна робота з пам'яттю.

Microsoft SQL Server — комерційна система управління базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-

SQL є реалізацією стандарту ANSI/ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних [6].

Головною перевагою СУБД Microsoft SQL Server є повна сумісність з продуктами від Microsoft.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Даний програмний продукт являє собою веб-застосування для пошуку місця для проведення конференції.

Для правильної роботи застосування до складу технічних засобів повинні входити наступні компоненти:

- комп’ютер, що має конфігурацію наведену нижче:
  1) процесор з тактовою частотою не нижче 1 ГГц;
  2) об’єм оперативної пам’яті не менше 2 Гб;
- встановлений один із перелічених нижче браузерів:
  1) Google Chrome;
  2) Apple Safari версії 2 або вище;
  3) Mozilla Firefox версії 1.5 або вище;
  4) Microsoft Internet Explorer версії 6 або вище;
  5) Opera 9 або вище.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Опишемо структуру діаграми класів.

Інтерфейс IService містить абстракцію поведінки характерної для всіх користувачів системи.

Інтерфейси IGuestService, IClientService, IHomeownerService наслідують інтерфейс IService та доповнюють його власними абстракціями.

Класи GuestService, ClientService, HomeownerService реалізують відповідні інтерфейси, визначаючи конкретну поведінку для кожної групи користувачів системи.

Абстрактний клас User містить в собі методи виклику відповідних функцій системи, характерні для всіх користувачів системи.

Клас Database є посередником між сервісом та базою даних. Агрегує в собі екземпляри класу Advertisement.

Схема структурна класів програмного забезпечення представлена в графічному матеріалі.

4.3.2 Діаграма послідовності

В графічному матеріалі представлена схема структурна послідовності для процесу створення оголошення на якій показані взаємодії об'єктів, упорядковані за часом їхнього прояву. Діаграма складається із трьох анонімних об’єктів: **«:Homeowner»**, **«:Database»** та **«:Advertisement»**.

Об’єкт **«:Database»** є посередником між користувачами сервісу та базою даних та інкапсулює запити(запис, видалення, фільтрація, виведення) до БД у своїх методах.

Об’єкт **«:Homeowner»** (орендодавець) створює об’єкт **«:Advertisement»**(оголошення). Потім він звертається до об’єкту **«:Database»** для збереження оголошення в БД. Після цього орендодавець

має можливість видалити своє оголошення з БД також через звернення до об’єкту **«:Database».**

4.3.3 Діаграма компонентів

В графічному матеріалі представлена схема структурна компонентів. Вона складається із компонентів: **«Account», «Advertisement», «GuestService», «ClientService», «HomeownerService», «Guest», «Client», «Homeowner» «Database», «Service»**.

Компонент **«Service»** надає інтерфейси для реалізації **«GuestService», «ClientService», «HomeownerService».**

Компонент **«Database»** агрегує **«Advertisement».**

Компонент **«Service»** використовує **«Database».**

4.3.4 Специфікація функцій

Функції класів програмного забезпечення наведені в таблицях 4.1 – 4.6:

Таблиця 4.1 – Функції класу HomeownerService

Назва	**Примітка**
Клас: HomeownerService – містить в собі функціонал для роботи з базою даних, передбачений роллю орендодавця	
GetAdvertisementByReservation (int reservationID)	Повертає оголошення по id бронювання
GetClientByReservation (int reservationID)	Повертає клієнта по id бронювання
GetReservation (int homeownerID)	Повертає бронювання по всіх оголошеннях, створених орендодавцем по id
GetAdvertisementByName (int homeownerID, string name)	Повертає оголошення по його імені
GetAdvertisements (int homeownerID)	Повертає всі оголошення створені орендодавцем з id
GetPersonalInformation (int homeownerID)	Отримання даних орендодавця по його id

Продовження таблиці 4.1

Назва	Примітка
UpdateDatabase()	Оновлення бази даних
DeleteAdvertisements (List<ModelAdvertisement> advertisements)	Видалення з бази даних переданих оголошень
SaveAdvertisement (ModelAdvertisement advertisement)	Зберегти в базі даних передане оголошення
GetReservationByDateAndNameAdver(int homeownerID, DateTime date, string name)	Повертає бронювання по конкретному оголошенню та даті

Таблиця 4.2 – Функції класу Homeowner

Назва	Примітка
Клас: Homeowner – викликає методи класу HomeownerService	
CallGetAdvertisementByReservation(int reservationID)	Виклик функції GetAdvertisementByReservation (int reservationID)
CallGetClientByReservation (int reservationID)	Виклик функції GetClientByReservation (int reservationID)
CallGetReservation()	Виклик функції GetReservation()
CallGetAdvertisementByName (string name)	Виклик функції GetReservation()
CallGetAdvertisement()	Виклик функції GetAdvertisement()
CallGetPersonalInformation()	Виклик функції GetPersonalInformation()
CallUpdateDatabase()	Виклик функції UpdateDatabase()
CallDeleteAdvertisements (List<ModelAdvertisement> advertisements)	Виклик функції DeleteAdvertisements (List<ModelAdvertisement> advertisements)
CallSaveAdvertisement (ModelAdvertisement advertisement)	Виклик функції SaveAdvertisement (ModelAdvertisement advertisement)

Продовження таблиці 4.2

Назва	Примітка
CallGetReservationByDateAndNameAdver(DateTime date, string name)	Виклик функції GetReservationByDateAndNameAdver(DateTime date, string name)

Таблиця 4.3 – Функції класу ClientService

Назва	Примітка
Клас: ClientService – містить в собі функціонал для роботи з базою даних, передбачений роллю клієнта	
GetAdvertisementByReservation (int reservationID)	Повертає оголошення по id резервації
GetReservationClient(int clientID)	Повертає бронювання, здійсненні конкретним клієнтом
AddReservation (ModelReservation reservation)	Додає бронювання у базу даних
GetClient(int clientID)	Повертає клієнта по його id
GetReservationByDateAndID(DateTime date, int advertisementID)	Повертає бронювання по конкретному оголошенню та даті
GetAllAdvertisements()	Повертає всі оголошення
GetAdvertisementByID(int id)	Повертає оголошення по його id
FiltrationByCity(string city, List<ModelAdvertisement> adver)	Повертає відфільтрований список оголошень по місту
FiltrationByPrice(int max, List<ModelAdvertisement> adver)	Повертає відфільтрований список оголошень по сумі погодинної оренди
FiltrationByRoominess(int max, List<ModelAdvertisement> adver)	Повертає відфільтрований список оголошень по місткості приміщення

Таблиця 4.4 – Функції класу Client

Назва	Примітка
Клас: Client – викликає методи класу ClientService	
CallGetAdvertisementByReservation(int reservationID)	Виклик функції GetAdvertisementByReservation(int reservationID)
CallGetReservationClient()	Виклик функції GetReservationClient()
CallAddReservation (ModelReservation reservation)	Виклик функції AddReservation (ModelReservation reservation)
CallGetClient()	Виклик функції GetClient()
CallGetReservationByDateAndID (DateTime date, int advertisementID)	Виклик функції GetReservationByDateAndID (DateTime date, int advertisementID)
CallGetAllAdvertisements()	Виклик функції GetAllAdvertisements()
CallGetAdvertisementByID(int id)	Виклик функції GetAdvertisementByID(int id)

Таблиця 4.5 – Функції класу GuestService

Назва	Примітка
Клас: GuestService – містить в собі функціонал для роботи з базою даних, передбачений роллю гостя	
GetAllAdvertisements()	Повертає всі оголошення
GetAllCities()	Повертає всі міста згадані в створених оголошеннях
FiltrationByCity(string city, List<ModelAdvertisement> adver)	Повертає відфільтрований список оголошень по місту
FiltrationByPrice(int max, List<ModelAdvertisement> adver)	Повертає відфільтрований список оголошень по сумі погодинної оренди
FiltrationByRoominess(int max, List<ModelAdvertisement> adver)	Повертає відфільтрований список оголошень по місткості приміщення

Таблиця 4.6 – Функції класу Guest

Назва	Примітка
Клас: Guest – викликає методи класу GuestService	
CallGetAllAdvertisements()	Виклик функції GetAllAdvertisements()
CallGetAllCities()	Виклик функції GetAllCities()

4.4 Опис звітів

На сторінці орендодавця доступні відомості про всі бронювання залів, розміщених ним. На рисунку 4.4 наведена їхня структура.

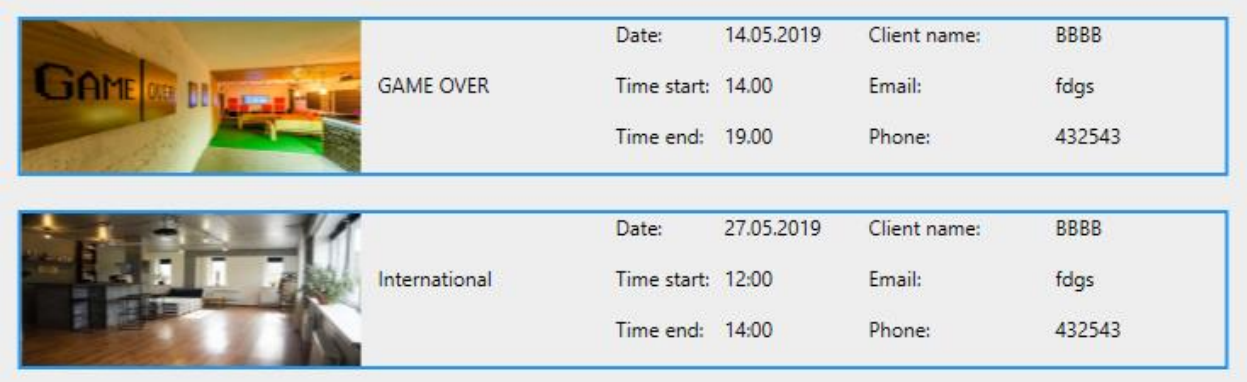


Рисунок 4.1 – Перелік бронювань

В першій колонці міститься фото, розміщене орендодавцем до відповідного оголошення; в другій – назва закладу; в третій згори до низу – дата резервування, початок та кінець інтервалу часу оренди приміщення; в четвертій згори до низу – ім’я, електронна адреса та номер телефону клієнта, який забронював даний зал.

Висновок до розділу

В даному розділі описані засоби розробки, використанні при створені застосування для дипломного проекти з їх характеристиками, особливостями та перевагами; спроектовано архітектуру програмного продукту за допомогою діаграми класів, послідовності та компонентів; описано функції

класів та представлено приклад звіту, який генерується розроблювальним застосуванням.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Функціонал системи передбачений для роботи з трьома типами осіб: гість, користувач та орендодавець. На рисунку 5.1 представлено початкову сторінку застосування після запуску програми. Дана сторінка являє собою весь функціонал передбачений для гостя. Гість може переглядати, фільтрувати та порівнювати оголошення.

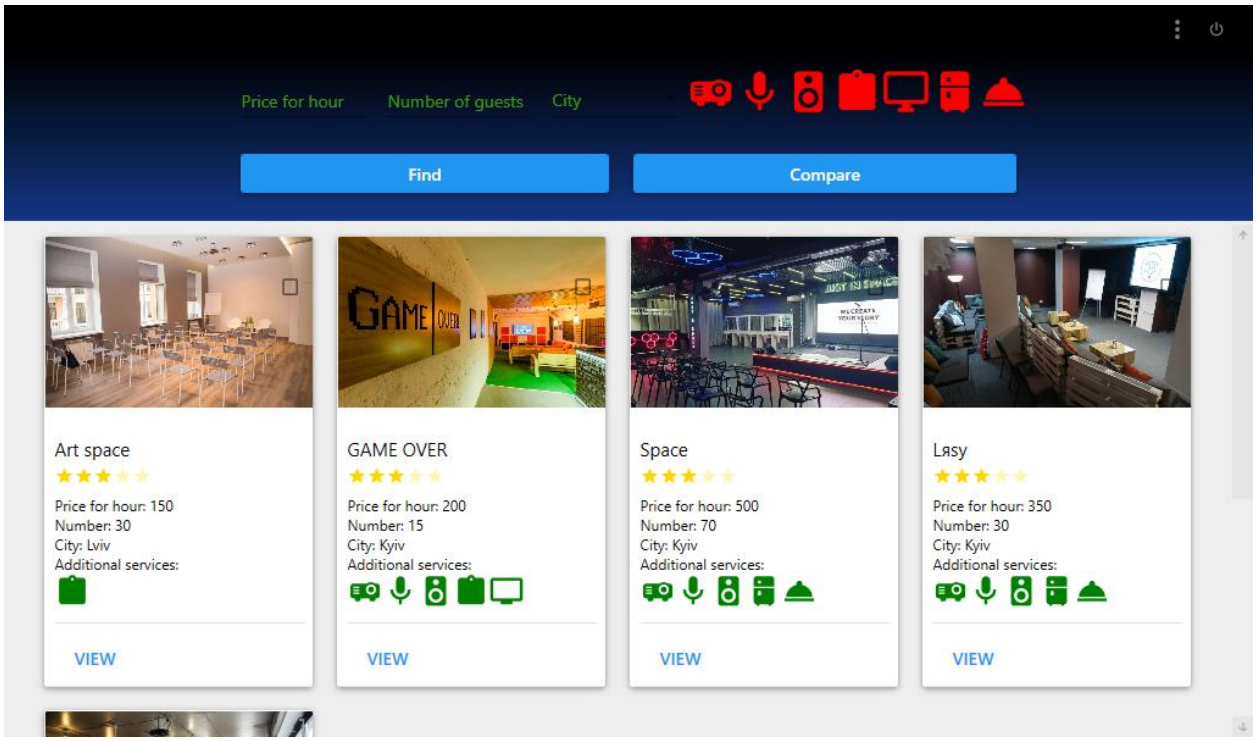


Рисунок 5.1 – Сторінка гостя

Для фільтрації оголошень достатньо ввести максимальну ціну погодинної оренди приміщення та кількість запрошених учасників, вибрати з випадаючого списку потрібне місто, виділити всі додаткові послуги, в яких ви маєте потребу та натиснути кнопку «Find». На рисунку 5.2 представлено результат фільтрації оголошень.

Рисунок 5.2 – Фільтрація оголошень

При натисканні кнопки «VIEW» відкривається детальна інформація про дане оголошення, яка включає в себе: назву локації; рейтинг, який підраховується на основі оцінок виставлених клієнтом; адреса; опис локації; розклад роботи; додаткові послуги(проектор, мікрофон, підсилювачі звуку, інтерактивна дошка, телевізор, холодильник, обслуговування в сфері харчування), використання яких включено в загальну суму оренди; сума погодинної оренди; максимальна місткість залу та правила, визначені орендодавцем. На рисунку 5.3 представлено детальний опис оголошення.

Рисунок 5.3 – Детальний опис оголошення

При виборі двох оголошень та натисканні кнопки «Compare» з’явиться порівняльна характеристика кожного оголошення з використанням всієї інформації представленої в детальному опису. На рисунку 5.4 представлено порівняння двох оголошень.

Рисунок 5.4 – Порівняння двох оголошень

У правому верхньому куті знаходиться кнопка при натисканні на яку, виводиться меню з відповідними пунктами: «Log in»(авторизація), «Registration»(реєстрація), «Exit»(вихід з системи). На рисунку 5.5 представлено відкрите меню.

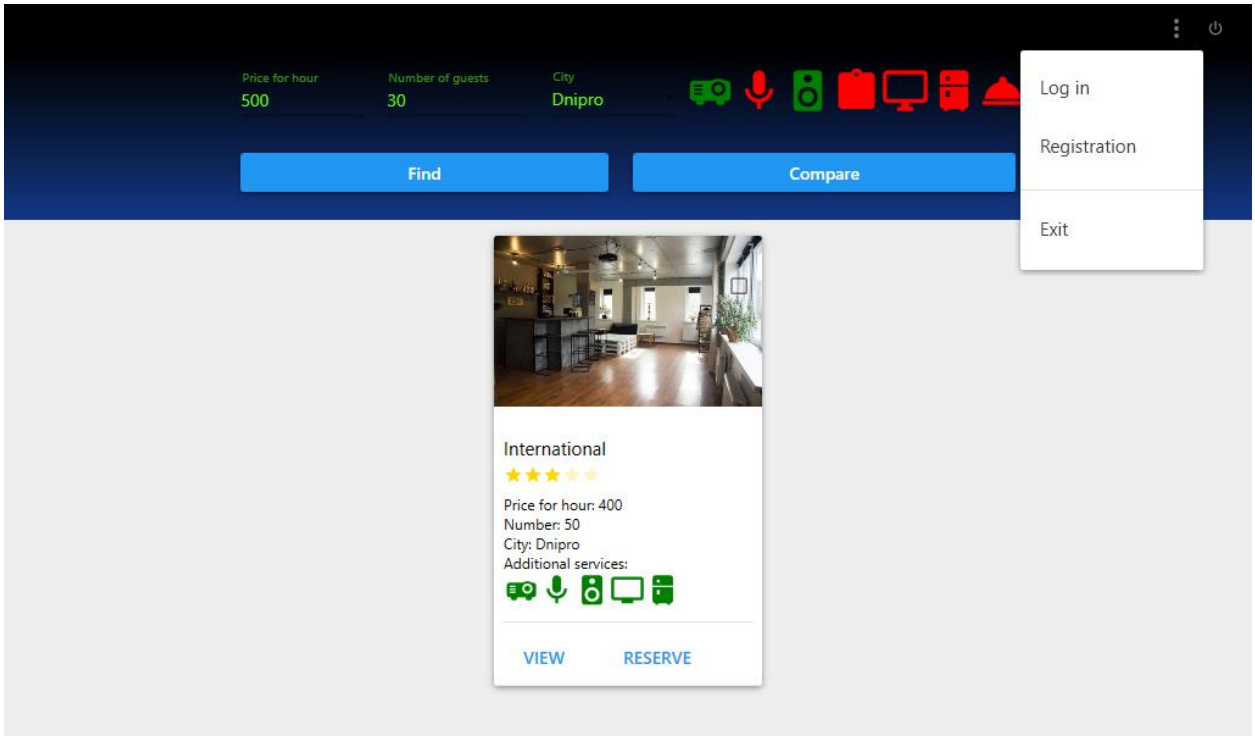


Рисунок 5.5 – Контекстне меню

При натисканні «Log in» відкривається окреме вікно, представлене на рисунку 5.6. В даному вікні потрібно ввести два обов’язкові поля(логін та пароль) та вибрати тип користувача(за замовчування - клієнт), після цього натиснути кнопку «LOGIN».

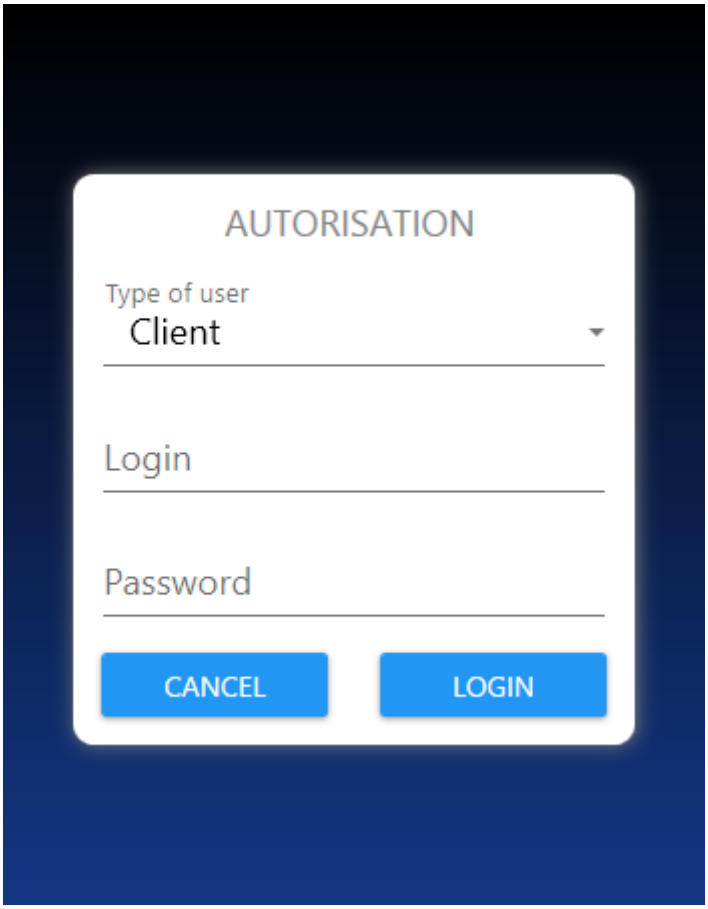


Рисунок 5.6 – Форма авторизації

При натисканні «Registration» відкривається окреме вікно, представлене на рисунку 5.7. В даному вікні всі поля є обов'язковими для введення. Загалом потрібно ввести: ім'я користувача, номер мобільного телефону, адресу електронної пошти, логін, пароль та повторення паролю. Адреса електронної пошти та номер мобільного телефону, введені під час реєстрації, будуть використанні при формуванні резервації локацій. Також потрібно вибрати тип користувача(за замовчування - клієнт).

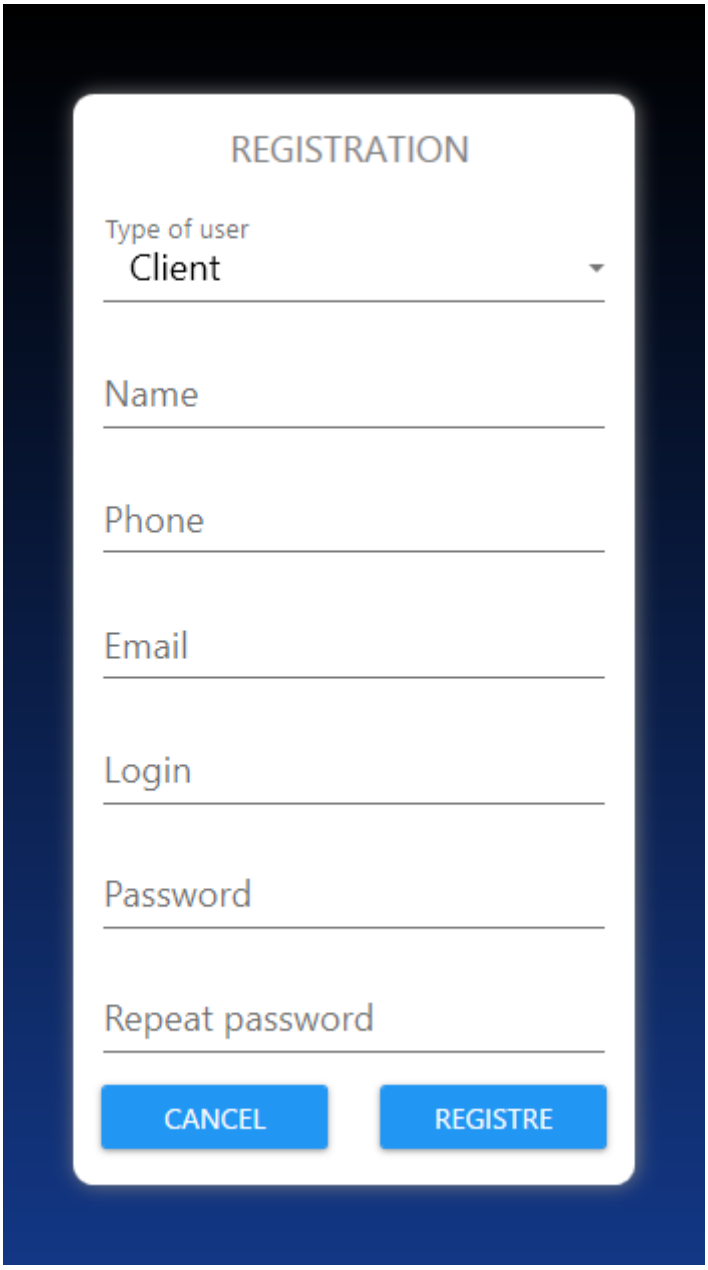


Рисунок 5.7 – Форма реєстрації

Після авторизації в ролі клієнта відкривається форма представлена на рисунку 5.8. Зліва міститься бокове меню, яке складається з наступних пунктів: «Advertisements»(оголошення), «Recommendation»(рекомендації), «My reservation»(мої бронювання), «My information»(моя інформація). При відкриті форми відкривається вкладка «Advertisements», в якій міститься функціонал аналогічний до гостя, за винятком, можливості резервування локації та можливості залишати коментар з оцінкою.

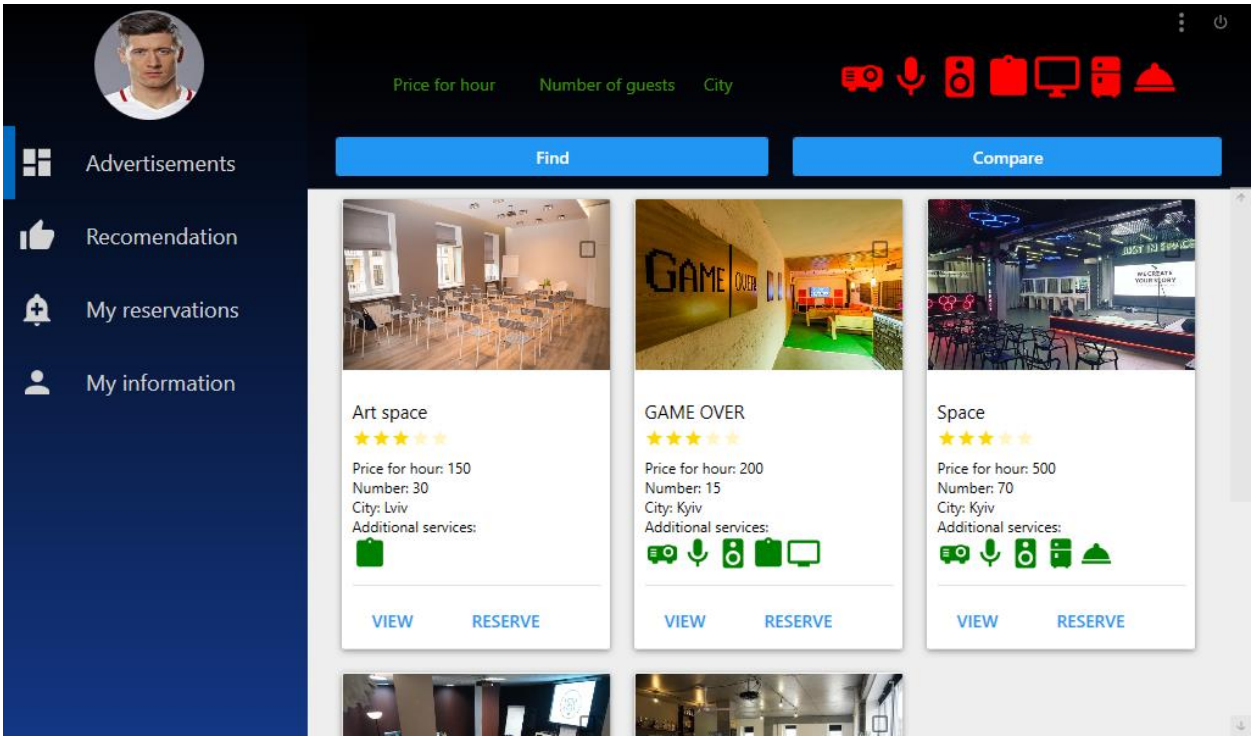


Рисунок 5.8 – Форма клієнта

При відкриті вкладки «Recomendation» з'являється перелік оголошень рекомендованих користувачу для перегляду. В даному переліку містяться оголошення, які з високою імовірністю зацікавлять користувача при пошуку місця для проведення конференції. Даний перелік щоденно оновлюється з можливістю як додавання нових так і видаленням старих рекомендацій. В даній вкладці доступний весь функціонал попередньої вкладки: фільтрація, перегляд та резервування локацій. Саме функціонал видачі рекомендацій і реалізовує алгоритм колаборативної фільтрації. На рисунку 5.9 представлено вкладку із рекомендаціями.

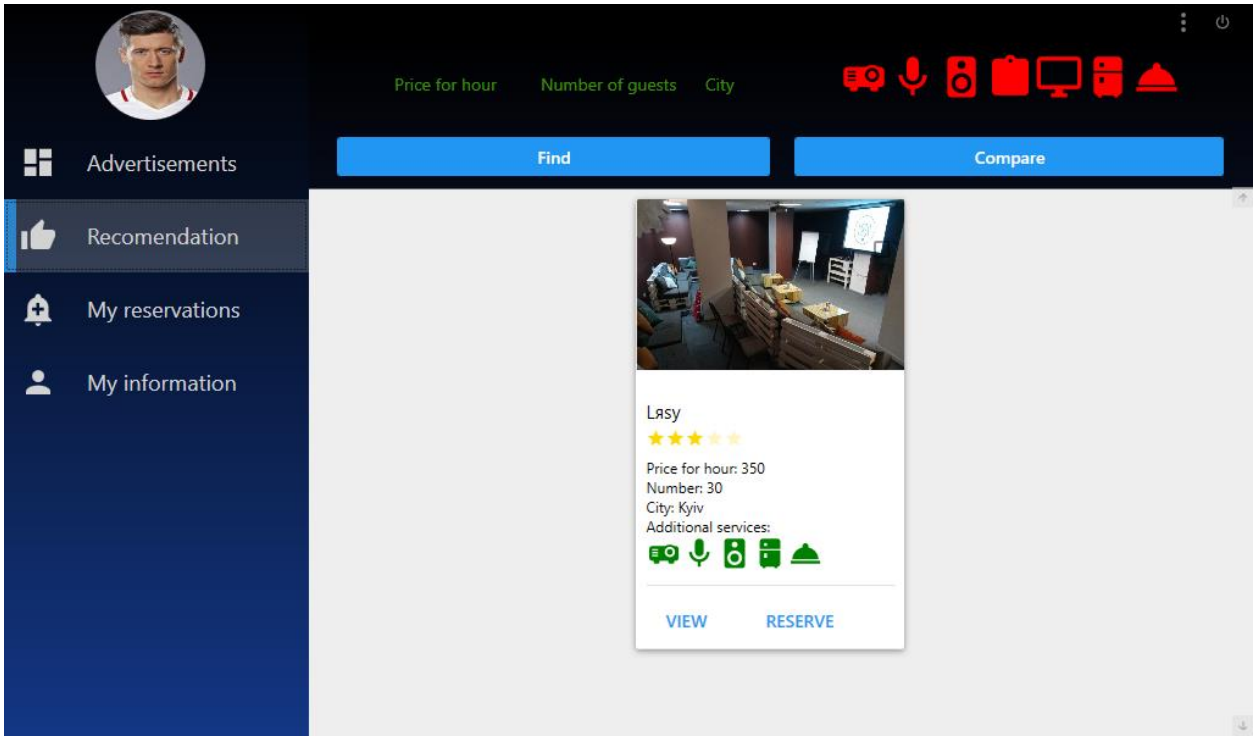


Рисунок 5.9 – Вкладка із рекомендаціями

При відкриті вкладки «My reservation» з'являється список всіх здійснених бронювань. Кожен елемент списку містить в собі фотографію та назву локації, дату та час резервування, кнопку «Cancel reservation», натиск якої відміняє бронювання. На рисунку 5.10 представлено вкладку із здійсненими бронюваннями.

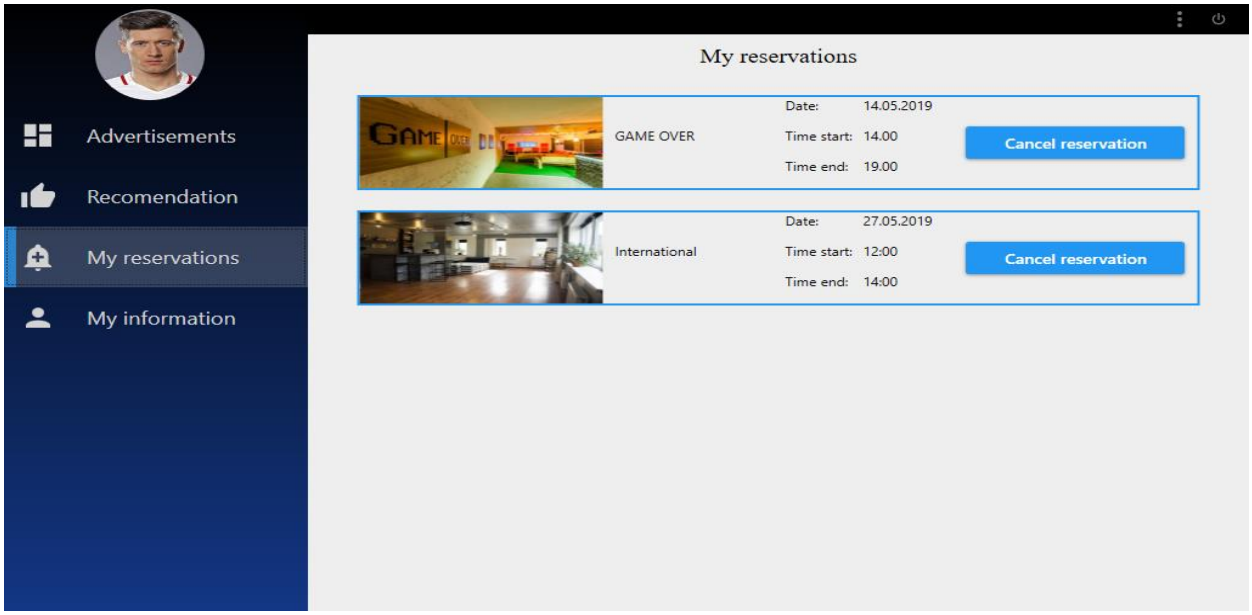


Рисунок 5.10 – Вкладка з бронюваннями

При відкриті вкладки «My information» з’являються персональні дані користувача. Користувач може модифікувати персональні дані, змінюючи фото, ім’я, електронну пошту, номер мобільного телефону, логін чи пароль. На рисунку 5.11 представлено вкладку із персональними даними користувача.

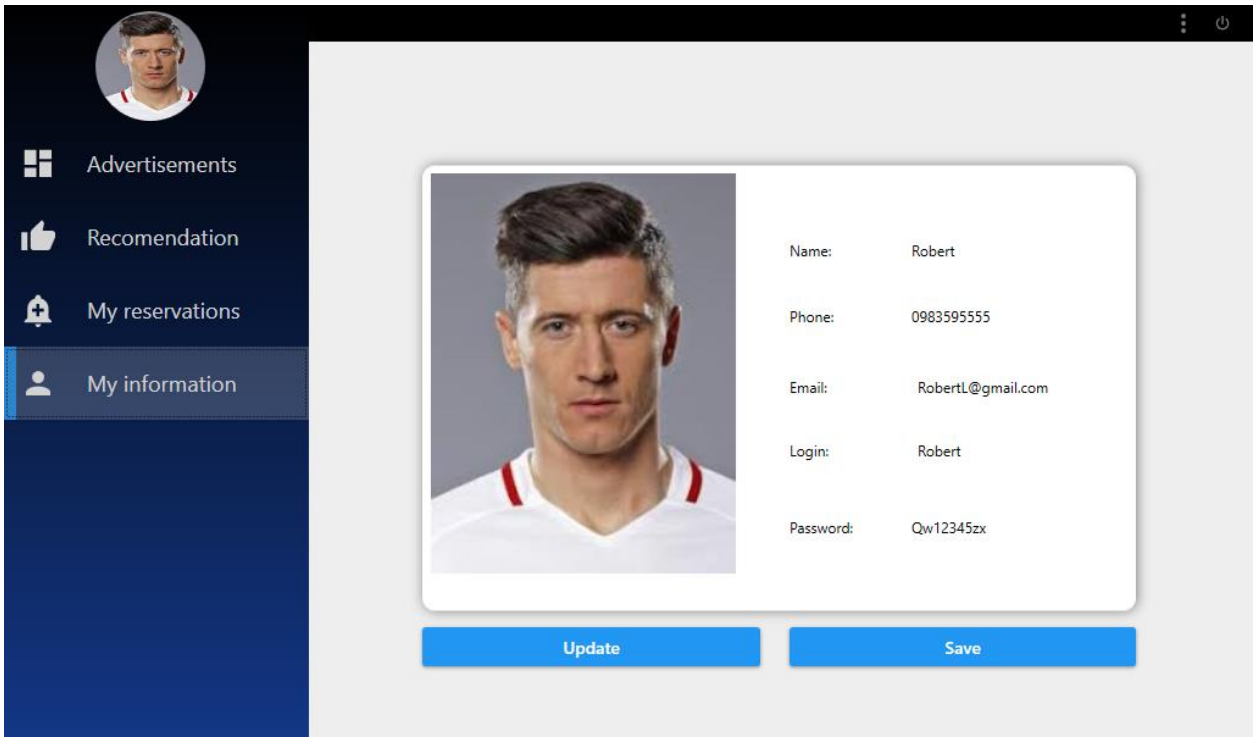


Рисунок 5.11 – Вкладка з персональними даними користувача

5.2 Випробування програмного продукту

В цьому підрозділі наведено опис тестів і порядок їх виконання для перевірки відповідності програмного забезпечення функціональним вимогам, представленим у технічному завданні на створення задачі контентної фільтрації для організації конференцій.

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій задачі вибору місця для організації конференції за рахунок колаборативної фільтрації вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В процесі тестування були перевірена уся функціональність системи. У наступних таблицях наведений перелік випробувань основних функціональних можливостей (таблиці 6.1 – 6.32).

Таблиця 6.1 – Реєстрація

Назва	**Перевірка функції «Реєстрація»**
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести у поле «Login» вже існуючий логін, у поле «Password» - унікальний пароль(кількість символів не менша семи), у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про вже існуючий логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.2 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - вже існуючий пароль, у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про вже існуючий логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.3 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - унікальний пароль(кількість символів менша семи), у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про занадто короткий пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.4 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - унікальний пароль(кількість символів не менша семи), у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, поле «Phone» залишити пустим. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про незаповненість всіх полів. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.5 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - унікальний пароль(кількість символів не менша семи), у поле «Name» - символьний рядок, поле «Email» залишити пустим, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про незаповненість всіх полів. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.6 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - унікальний пароль(кількість символів не менша семи), поле «Name» залишити пустим, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про незаповненість всіх полів. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.7 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», поле «Password» залишити пустим, у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про незаповненість всіх полів. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.8 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Поле «Login» залишити пустим, у поле «Password» ввести унікальний пароль(кількість символів не менша семи), у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Повідомлення про незаповненість всіх полів. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Реєстрація»

Таблиця 6.9 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - унікальний пароль(кількість символів не менша семи), у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «Registration»
Очікуваний результат:	Відкрита сторінка клієнта
Стан системи після проведення випробувань:	Відкрита сторінка клієнта

Таблиця 6.10 – Реєстрація

Назва	Перевірка функції «Реєстрація»
Початковий стан системи	Відкрита форма «Реєстрація»
Вхідні данні:	Логін, пароль, ім’я, електронна пошта, мобільний телефон, тип користувача
Схема проведення тесту:	Ввести унікальний логін у поле «Login», у поле «Password» - унікальний пароль(кількість символів не менша семи), у поле «Name» - символьний рядок, у поле «Email» - коректна електронна адреса, у поле «Phone» - коректний номер телефону. Вибрати тип користувача - орендодавець. Натиснути кнопку «Registration»
Очікуваний результат:	Відкрита сторінка орендодавця
Стан системи після проведення випробувань:	Відкрита сторінка орендодавця

Таблиця 6.11 – Авторизація

Назва	Перевірка функції «Авторизація»
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача
Схема проведення тесту:	Ввести неіснуючий логін у поле «Login», у поле «Password» - існуючий пароль. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «LOGIN»
Очікуваний результат:	Повідомлення про некоректний логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Авторизація»

Таблиця 6.12 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача
Схема проведення тесту:	Ввести існуючий в системі логін у поле «Login», у поле «Password» - неіснуючий пароль. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «LOGIN»
Очікуваний результат:	Повідомлення про некоректний логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Авторизація»

Таблиця 6.13 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача
Схема проведення тесту:	Ввести неіснуючий в системі логін у поле «Login», у поле «Password» - неіснуючий пароль. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «LOGIN»
Очікуваний результат:	Повідомлення про некоректний логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Авторизація»

Таблиця 6.14 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача

Продовження таблиці 6.14

Схема проведення тесту:	Ввести логін, одного із клієнтів, у поле «Login», у поле «Password» - пароль іншого клієнта. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «LOGIN»
Очікуваний результат:	Повідомлення про некоректний логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Авторизація»

Таблиця 6.15 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача
Схема проведення тесту:	Ввести логін, одного із клієнтів, у поле «Login», у поле «Password» - пароль одного із орендодавців. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «LOGIN»
Очікуваний результат:	Повідомлення про некоректний логін або пароль. Вхід на сторінку клієнта не виконано
Стан системи після проведення випробувань:	Відкрита форма «Авторизація»

Таблиця 6.16 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача
Схема проведення тесту:	Ввести існуючий логін клієнта у поле «Login», у поле «Password» - існуючий пароль клієнта. Залишити за замовчуванням тип користувача(клієнт). Натиснути кнопку «LOGIN»
Очікуваний результат:	Відкрита сторінка клієнта

Продовження таблиці 6.16

Стан системи після проведення випробувань:	Відкрита сторінка клієнта

Таблиця 6.17 – Авторизація

Назва	**Перевірка функції «Авторизація»**
Початковий стан системи	Відкрита форма «Авторизація»
Вхідні данні:	Логін, пароль, тип користувача
Схема проведення тесту:	Ввести існуючий логін орендодавця у поле «Login», у поле «Password» - існуючий пароль орендодавця. Вибрати тип користувача - орендодавець. Натиснути кнопку «LOGIN»
Очікуваний результат:	Відкрита сторінка орендодавця
Стан системи після проведення випробувань:	Відкрита сторінка орендодавця

Таблиця 6.18 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Ввести максимальну ціну погодинної оренди у поле «Price for hour», у поле «Number of guests» - мінімальну місткість приміщення. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення списку оголошень, які задовольняють введеній ціні, місткості та вибраним місту та додатковим послугам
Стан системи після проведення випробувань:	Виведення списку оголошень, які задовольняють введеній ціні, місткості та вибраним місту та додатковим послугам

Таблиця 6.19 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Залишити поле «Price for hour» пустим, ввести у поле «Number of guests» мінімальну місткість приміщення. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення списку оголошень, які задовольняють введеній місткості та вибраним місту та додатковим послугам
Стан системи після проведення випробувань:	Виведення списку оголошень, які задовольняють введеній місткості та вибраним місту та додатковим послугам

Таблиця 6.20 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Залишити поле «Price for hour» та «Number of guests» пустими. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення списку оголошень, які задовольняють вибраним місту та додатковим послугам
Стан системи після проведення випробувань:	Виведення списку оголошень, які задовольняють вибраним місту та додатковим послугам

Таблиця 6.21 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Залишити поле «Price for hour» та «Number of guests» пустими. Вибрати лише додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення списку оголошень, які задовольняють вибраним додатковим послугам
Стан системи після проведення випробувань:	Виведення списку оголошень, які задовольняють вибраним додатковим послугам

Таблиця 6.22 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Залишити поле «Price for hour» та «Number of guests» пустими. Натиснути кнопку «Find»
Очікуваний результат:	Виведення всіх оголошень в системі
Стан системи після проведення випробувань:	Виведення всіх оголошень в системі

Таблиця 6.23 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги

Продовження таблиці 6.23

Схема проведення тесту:	Ввести символьний рядок у поле «Price for hour», у поле «Number of guests» - мінімальну місткість приміщення. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення повідомлення про неправильний вхідний формат даних у полі «Price for hour». Фільтрацію оголошень не виконано
Стан системи після проведення випробувань:	Відкрита сторінка гостя

Таблиця 6.24 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Ввести максимальну ціну погодинної оренди у поле «Price for hour», у поле «Number of guests» - символьний рядок. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення повідомлення про неправильний вхідний формат даних у полі «Number of guests». Фільтрацію оголошень не виконано
Стан системи після проведення випробувань:	Відкрита сторінка гостя

Таблиця 6.25 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги

Продовження таблиці 6.25

Схема проведення тесту:	Ввести від’ємне значення у поле «Price for hour», у поле «Number of guests» - мінімальну місткість приміщення. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення повідомлення про неправильний вхідний формат даних у полі «Price for hour». Фільтрацію оголошень не виконано
Стан системи після проведення випробувань:	Відкрита сторінка гостя

Таблиця 6.26 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Ввести нуль у поле «Price for hour», у поле «Number of guests» - мінімальну місткість приміщення. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення повідомлення про неправильний вхідний формат даних у полі «Price for hour». Фільтрацію оголошень не виконано
Стан системи після проведення випробувань:	Відкрита сторінка гостя

Таблиця 6.27 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Ввести максимальну ціну погодинної оренди у поле «Price for hour», у поле «Number of guests» - від’ємне значення. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення повідомлення про неправильний вхідний формат даних у полі «Number of guests». Фільтрацію оголошень не виконано
Стан системи після проведення випробувань:	Відкрита сторінка гостя

Таблиця 6.28 – Фільтрація

Назва	**Перевірка функції «Фільтрація»**
Початковий стан системи	Відкрита сторінка гостя
Вхідні данні:	Максимальна ціна погодинної оренди, мінімальна місткість залу, місто, додаткові послуги
Схема проведення тесту:	Ввести максимальну ціну погодинної оренди у поле «Price for hour», у поле «Number of guests» - нуль. Вибрати місто та додаткові послуги. Натиснути кнопку «Find»
Очікуваний результат:	Виведення повідомлення про неправильний вхідний формат даних у полі «Number of guests». Фільтрацію оголошень не виконано
Стан системи після проведення випробувань:	Відкрита сторінка гостя

Таблиця 6.29 – Бронювання

Назва	**Перевірка функції «Бронювання»**
Початковий стан системи	Відкрита форма бронювання
Вхідні данні:	Дата, початок та закінчення часу резервування
Схема проведення тесту:	Вибрати початок часу резервування більшим ніж час закінчення резервування. Вибрати дату. Натиснути кнопку «Reserve»
Очікуваний результат:	Виведення повідомлення про від'ємну тривалість інтервалу часу. Бронювання залу не виконано
Стан системи після проведення випробувань:	Відкрита форма бронювання

Таблиця 6.30 – Бронювання

Назва	**Перевірка функції «Бронювання»**
Початковий стан системи	Відкрита форма бронювання
Вхідні данні:	Дата, початок та закінчення часу резервування
Схема проведення тесту:	Вибрати початок часу резервування меншим ніж час закінчення резервування. Вибрати дату, яка вже минула. Натиснути кнопку «Reserve»
Очікуваний результат:	Виведення повідомлення про неможливість бронювання по даній даті. Бронювання залу не виконано
Стан системи після проведення випробувань:	Відкрита форма бронювання

Таблиця 6.31 – Бронювання

Назва	Перевірка функції «Бронювання»
Початковий стан системи	Відкрита форма бронювання
Вхідні данні:	Дата, початок та закінчення часу резервування
Схема проведення тесту:	Вибрати початок часу резервування меншим ніж час закінчення резервування, даний інтервал має перетин з вже забронюваними. Вибрати дату. Натиснути кнопку «Reserve»
Очікуваний результат:	Виведення повідомлення про існування бронювань з таким інтервалом часу. Бронювання залу не виконано
Стан системи після проведення випробувань:	Відкрита форма бронювання

Таблиця 6.32 – Бронювання

Назва	Перевірка функції «Бронювання»
Початковий стан системи	Відкрита форма бронювання
Вхідні данні:	Дата, початок та закінчення часу резервування
Схема проведення тесту:	Вибрати початок часу резервування меншим ніж час закінчення резервування, даний інтервал не має перетину з вже забронюваними. Вибрати дату. Натиснути кнопку «Reserve»
Очікуваний результат:	Бронювання залу виконано
Стан системи після проведення випробувань:	Відкрита сторінка клієнта

Висновок до розділу

В даному розділі наведено керівництво користувача з описом всіх екранних форм та їх поясненнями. Крім того складено 32 тест кейси для перевірки коректності роботи програми. Загалом були перевірені функції реєстрації, авторизації, фільтрації оголошень та бронювання локацій.

ЗАГАЛЬНІ ВИСНОВКИ

В даній дипломній роботі реалізовано алгоритм колаборативної фільтрації фільтрації для реалізації функціоналу видачі рекомендацій клієнтам, розроблювального застосування. Крім того в програмному продукті реалізовані функції реєстрації та авторизації користувачів, перегляду, оновленя, додавання, видалення та фільтрації оголошень, можливість бронювати зал на конкретну дату та інтервал часу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sarwar B. Item-Based Collaborative Filtering Recommendation Algorithms / Sarwar B., Karypis G., Konstan J., and Riedl J. // Hong Kong WWW10 Conference : «WWW10», 20-25 January 2001, Hong Kong : materials.–Hong Kong : WWW10 Press, 2001. –С.285-289.
2. [Електронний ресурс] Режим доступу: https://docs.microsoft.com/ru-ru/visualstudio/designers/introduction-to-wpf?view=vs-2019
3. [Електронний ресурс] Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/advanced/xaml-overview-wpf
4. [Електронний ресурс] Режим доступу: https://metanit.com/sharp/entityframework/1.1.php
5. [Електронний ресурс] Режим доступу: https://damp.biz/mova-programuvannya-c-si-sharp-entsiklopediya/
6. [Електронний ресурс] Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server

Додаток А

Тексти програмного коду

Система пошуку місць для проведення конференцій

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 арк, 244 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Клас ModelAdvertisement:

```
public class ModelAdvertisement
{
        [Key]
        public int AdvertisementId { get; set; }
        public ModelHomeowner HomeownerId { get; set; }
        public byte[] Image { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public int Roominess { get; set; }
        public float Price { get; set; }
        public string City { get; set; }
        public bool TV { get; set; }
        public bool Speakers { get; set; }
        public bool Microphone { get; set; }
        public bool Projector { get; set; }
        public bool FlipChart { get; set; }
        public bool Refrigerator { get; set; }
        public bool Kitchen { get; set; }
        public string Description { get; set; }
        public string Rules { get; set; }
        public string Shedule { get; set; }
}
```

Клас ModelClient:

```
public class ModelClient
{
        [Key]
        public int ClientId { get; set; }
        public string Name { get; set; }
        public string Mobile { get; set; }
        public string Email { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }
}
```

Клас ModelHomeowner:

```
public class ModelHomeowner
{
        [Key]
        public int HomeownerId { get; set; }
        public string Name { get; set; }
        public string Mobile { get; set; }
        public string Email { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }
}
```

Клас ModelRating:

```
public class ModelRating
{
        [Key]
        public int RatingId { get; set; }
        public ModelAdvertisement AdvertisementId { get; set; }
        public ModelClient ClientId { get; set; }
        public int Mark { get; set; }
        public string Comment { get; set; }
 }
```

Клас ModelRecommendation:

```
public class ModelRecommendation
{
        [Key]
        public int RecommendationId { get; set; }
        public ModelAdvertisement AdvertisementId { get; set; }
        public ModelClient ClientId { get; set; }
}
```

Клас ModelReservation:

```
public class ModelReservation
{
        [Key]
        public int ReservationId { get; set; }
        public ModelAdvertisement AdvertisementId { get; set; }
        public ModelClient ClientId { get; set; }
        public DateTime Date { get; set; }
        public string Time { get; set; }
}
```

Клас Context:

```
public class Context : DbContext
{
        public Context() : base("MyConnection") { }

        public DbSet<ModelClient> Clients { get; set; }
        public DbSet<ModelHomeowner> Homeowners { get; set; }
        public DbSet<ModelAdvertisement> Advertisements { get; set; }
        public DbSet<ModelRating> Ratings { get; set; }
        public DbSet<ModelRecommendation> Recommendations { get; set; }
        public DbSet<ModelReservation> Reservations { get; set; }
}
```

Клас ClientService:

```
public class ClientService : IClientService
    {
        private Context db;

        public ClientService()
        {
            db = new Context();
        }

        public ModelAdvertisement GetAdvertisementByReservation(int reservationID)
        {
            int id = db.Reservations.Where(t => t.ReservationId ==
reservationID).Select(t => t.AdvertisementId.AdvertisementId).First();

            return db.Advertisements.Where(t => t.AdvertisementId == id).First();
        }
        public List<ModelReservation> GetReservationClient(int clientID)
        {
            return db.Reservations.Where(t => t.ClientId.ClientId ==
clientID).ToList<ModelReservation>();
        }

        public void AddReservation(ModelReservation reservation)
        {
            db.Reservations.Add(reservation);
            db.SaveChanges();
        }

        public ModelClient GetClient(int clientID)
        {
            return db.Clients.Where(t => t.ClientId == clientID).First();
        }

        public List<ModelReservation> GetReservationByDateAndID(DateTime date, int
advertisementID)
        {
            return db.Reservations.Where(t => t.AdvertisementId.AdvertisementId ==
advertisementID).Where(t => t.Date == date).ToList<ModelReservation>();
        }

        public List<ModelAdvertisement> GetAllAdvertisements()
        {
            return db.Advertisements.ToList<ModelAdvertisement>();
        }

        public ModelAdvertisement GetAdvertisementByID(int id)
        {
            return db.Advertisements.Where(t => t.AdvertisementId == id).First();
        }

        public List<ModelAdvertisement> FiltrationAdvertisements(int max_price, int
max_guests, string city, bool projector, bool microphone, bool speakers, bool flipchart,
bool tv, bool refrigerator, bool kitchen)
        {
            List<ModelAdvertisement> adver = GetAllAdvertisements();

            if(max_price > 0)
            {
                adver = FiltrationByPrice(max_price, adver);
            }
```

```
            if(max_guests > 0)
            {
                adver = FiltrationByRoominess(max_guests, adver);
            }

            if(city != "")
            {
                adver = FiltrationByCity(city, adver);
            }

            adver = FiltrationByAdditionalServices(projector, microphone, speakers,
flipchart, tv, refrigerator, kitchen, adver);

            return adver;
        }

        public List<ModelAdvertisement> FiltrationByCity(string city,
List<ModelAdvertisement> adver)
        {
            return adver.Where(t => t.City == city).ToList<ModelAdvertisement>();
        }

        public List<ModelAdvertisement> FiltrationByPrice(int max,
List<ModelAdvertisement> adver)
        {
            return adver.Where(t => t.Price <= max).ToList<ModelAdvertisement>();
        }

        public List<ModelAdvertisement> FiltrationByRoominess(int max,
List<ModelAdvertisement> adver)
        {
            return adver.Where(t => t.Roominess >= max).ToList<ModelAdvertisement>();
        }

        public List<ModelAdvertisement> FiltrationByAdditionalServices(bool projector,
bool microphone, bool speakers, bool flipchart, bool tv, bool refrigerator, bool kitchen,
List<ModelAdvertisement> adver)
        {
            if(projector)
            {
                adver = adver.Where(t => t.Projector).ToList<ModelAdvertisement>();
            }

            if (microphone)
            {
                adver = adver.Where(t => t.Microphone).ToList<ModelAdvertisement>();
            }

            if (speakers)
            {
                adver = adver.Where(t => t.Speakers).ToList<ModelAdvertisement>();
            }

            if (flipchart)
            {
                adver = adver.Where(t => t.FlipChart).ToList<ModelAdvertisement>();
            }

            if (tv)
            {
                adver = adver.Where(t => t.TV).ToList<ModelAdvertisement>();
            }
```

```
            if (refrigerator)
            {
                adver = adver.Where(t => t.Refrigerator).ToList<ModelAdvertisement>();
            }

            if (kitchen)
            {
                adver = adver.Where(t => t.Kitchen).ToList<ModelAdvertisement>();
            }

            return adver;
        }
    }
```

Клас Client:

```
public class Client
    {
        public int clientID;
        public ClientService cs;

        public Client(int id)
        {
            clientID = id;
            cs = new ClientService();
        }

        public ModelAdvertisement CallGetAdvertisementByReservation(int reservationID)
        {
            return cs.GetAdvertisementByReservation(reservationID);
        }

        public List<ModelReservation> CallGetReservationClient()
        {
            return cs.GetReservationClient(clientID);
        }

        public ModelClient CallGetClient()
        {
            return cs.GetClient(clientID);
        }

        public ModelAdvertisement CallGetAdvertisementByID(int id)
        {
            return cs.GetAdvertisementByID(id);
        }

        public void CallAddReservation(ModelReservation reservation)
        {
            cs.AddReservation(reservation);
        }

        public List<ModelReservation> CallGetReservationByDateAndID(DateTime date, int
advertisementID)
        {
            return cs.GetReservationByDateAndID(date, advertisementID);
        }
```

```
        public List<ModelAdvertisement> CallGetAllAdvertisements()
        {
            return cs.GetAllAdvertisements();
        }
        public List<ModelAdvertisement> CallFiltrationAdvertisements(int max_price, int
max_guests, string city, bool projector, bool microphone, bool speakers, bool flipchart,
bool tv, bool refrigerator, bool kitchen)
        {
            return cs.FiltrationAdvertisements(max_price, max_guests, city, projector,
microphone, speakers, flipchart, tv, refrigerator, kitchen);
        }
    }
```

Клас GuestService:

```
class GuestService : IGuestService
    {
        private Context db;

        public GuestService()
        {
            db = new Context();
        }

        public List<ModelAdvertisement> GetAllAdvertisements()
        {
            return db.Advertisements.ToList<ModelAdvertisement>();
        }

        public List<string> GetAllCities()
        {
            return db.Advertisements.Where(t => t.City != "").Select(t =>
t.City).Distinct().ToList<string>();
        }

        public List<ModelAdvertisement> FiltrationAdvertisements(int max_price, int
max_guests, string city, bool projector, bool microphone, bool speakers, bool flipchart,
bool tv, bool refrigerator, bool kitchen)
        {
            List<ModelAdvertisement> adver = GetAllAdvertisements();

            if (max_price > 0)
            {
                adver = FiltrationByPrice(max_price, adver);
            }

            if (max_guests > 0)
            {
                adver = FiltrationByRoominess(max_guests, adver);
            }

            if (city != "")
            {
                adver = FiltrationByCity(city, adver);
            }

            adver = FiltrationByAdditionalServices(projector, microphone, speakers,
flipchart, tv, refrigerator, kitchen, adver);

            return adver;
        }
```

```
        public List<ModelAdvertisement> FiltrationByCity(string city,
List<ModelAdvertisement> adver)
        {
            return adver.Where(t => t.City == city).ToList<ModelAdvertisement>();
        }

        public List<ModelAdvertisement> FiltrationByPrice(int max,
List<ModelAdvertisement> adver)
        {
            return adver.Where(t => t.Price <= max).ToList<ModelAdvertisement>();
        }

        public List<ModelAdvertisement> FiltrationByRoominess(int max,
List<ModelAdvertisement> adver)
        {
            return adver.Where(t => t.Roominess >= max).ToList<ModelAdvertisement>();
        }

        public List<ModelAdvertisement> FiltrationByAdditionalServices(bool projector,
bool microphone, bool speakers, bool flipchart, bool tv, bool refrigerator, bool kitchen,
List<ModelAdvertisement> adver)
        {
            if (projector)
            {
                adver = adver.Where(t => t.Projector).ToList<ModelAdvertisement>();
            }

            if (microphone)
            {
                adver = adver.Where(t => t.Microphone).ToList<ModelAdvertisement>();
            }

            if (speakers)
            {
                adver = adver.Where(t => t.Speakers).ToList<ModelAdvertisement>();
            }

            if (flipchart)
            {
                adver = adver.Where(t => t.FlipChart).ToList<ModelAdvertisement>();
            }

            if (tv)
            {
                adver = adver.Where(t => t.TV).ToList<ModelAdvertisement>();
            }

            if (refrigerator)
            {
                adver = adver.Where(t => t.Refrigerator).ToList<ModelAdvertisement>();
            }

            if (kitchen)
            {
                adver = adver.Where(t => t.Kitchen).ToList<ModelAdvertisement>();
            }

            return adver;
        }
    }
```

Клас Guest:

```
class Guest
    {
        public GuestService gs;

        public Guest()
        {
            gs = new GuestService();
        }

        public List<string> CallGetAllCities()
        {
            return gs.GetAllCities();
        }

        public List<ModelAdvertisement> CallGetAllAdvertisements()
        {
            return gs.GetAllAdvertisements();
        }

        public List<ModelAdvertisement> CallFiltrationAdvertisements(int max_price, int
max_guests, string city, bool projector, bool microphone, bool speakers, bool flipchart,
bool tv, bool refrigerator, bool kitchen)
        {
            return gs.FiltrationAdvertisements(max_price, max_guests, city, projector,
microphone, speakers, flipchart, tv, refrigerator, kitchen);
        }

    }
```

Клас HomeownerService:

```
public class HomeownerService : IHomeownerService
    {
        private Context db;

        public HomeownerService()
        {
            db = new Context();
        }

        public ModelAdvertisement GetAdvertisementByReservation(int reservationID)
        {
            int id = db.Reservations.Where(t => t.ReservationId ==
reservationID).Select(t => t.AdvertisementId.AdvertisementId).First();

            return db.Advertisements.Where(t => t.AdvertisementId == id).First();
        }

        public ModelClient GetClientByReservation(int reservationID)
        {
            int id = db.Reservations.Where(t => t.ReservationId ==
reservationID).Select(t => t.ClientId.ClientId).First();

            return db.Clients.Where(t => t.ClientId == id).First();
        }
```

```
        public List<ModelReservation> GetReservation(int homeownerID)
        {
            List<ModelAdvertisement> advertisements = GetAdvertisements(homeownerID);
            List<ModelReservation> modelReservation = new List<ModelReservation>();

            var reservation = from r in db.Reservations.ToList<ModelReservation>()
                              join a in advertisements on r.AdvertisementId equals a
                              select new
                              {
                                  ReservationId = r.ReservationId,
                                  AdvertisementId = r.AdvertisementId,
                                  ClientId = r.ClientId,
                                  Date = r.Date,
                                  Time = r.Time
                              };

            foreach (var r in reservation)
            {
                ModelReservation m = new ModelReservation();
                m.ReservationId = r.ReservationId;
                m.AdvertisementId = r.AdvertisementId;
                m.ClientId = r.ClientId;
                m.Date = r.Date;
                m.Time = r.Time;
                modelReservation.Add(m);
            }

            return modelReservation;
        }

        public ModelAdvertisement GetAdvertisementByName(int homeownerID, string name)
        {
            return db.Advertisements.Where(t => t.HomeownerId.HomeownerId ==
homeownerID).Where(t => t.Name == name).First();
        }

        public List<ModelReservation> GetReservationByDateAndNameAdver(int homeownerID,
DateTime date, string name)
        {
            List<ModelReservation> modelReservation = GetReservation(homeownerID);

            return modelReservation.Where(t => t.Date == date).Where(t =>
t.AdvertisementId.Name == name).ToList<ModelReservation>();
        }

        public List<ModelReservation> GetReservationByAdvertisement(int homeownerID,
ModelAdvertisement advertisement)
        {
            List<ModelReservation> modelReservation = GetReservation(homeownerID);

            return modelReservation.Where(t => t.AdvertisementId ==
advertisement).ToList<ModelReservation>();
        }

        public List<ModelAdvertisement> GetAdvertisements(int homeownerID)
        {
            return db.Advertisements.Where(t => t.HomeownerId.HomeownerId ==
homeownerID).ToList<ModelAdvertisement>();
        }

        public ModelHomeowner GetPersonalInformation(int homeownerID)
        {
            return db.Homeowners.Where(t => t.HomeownerId == homeownerID).First();
        }
```

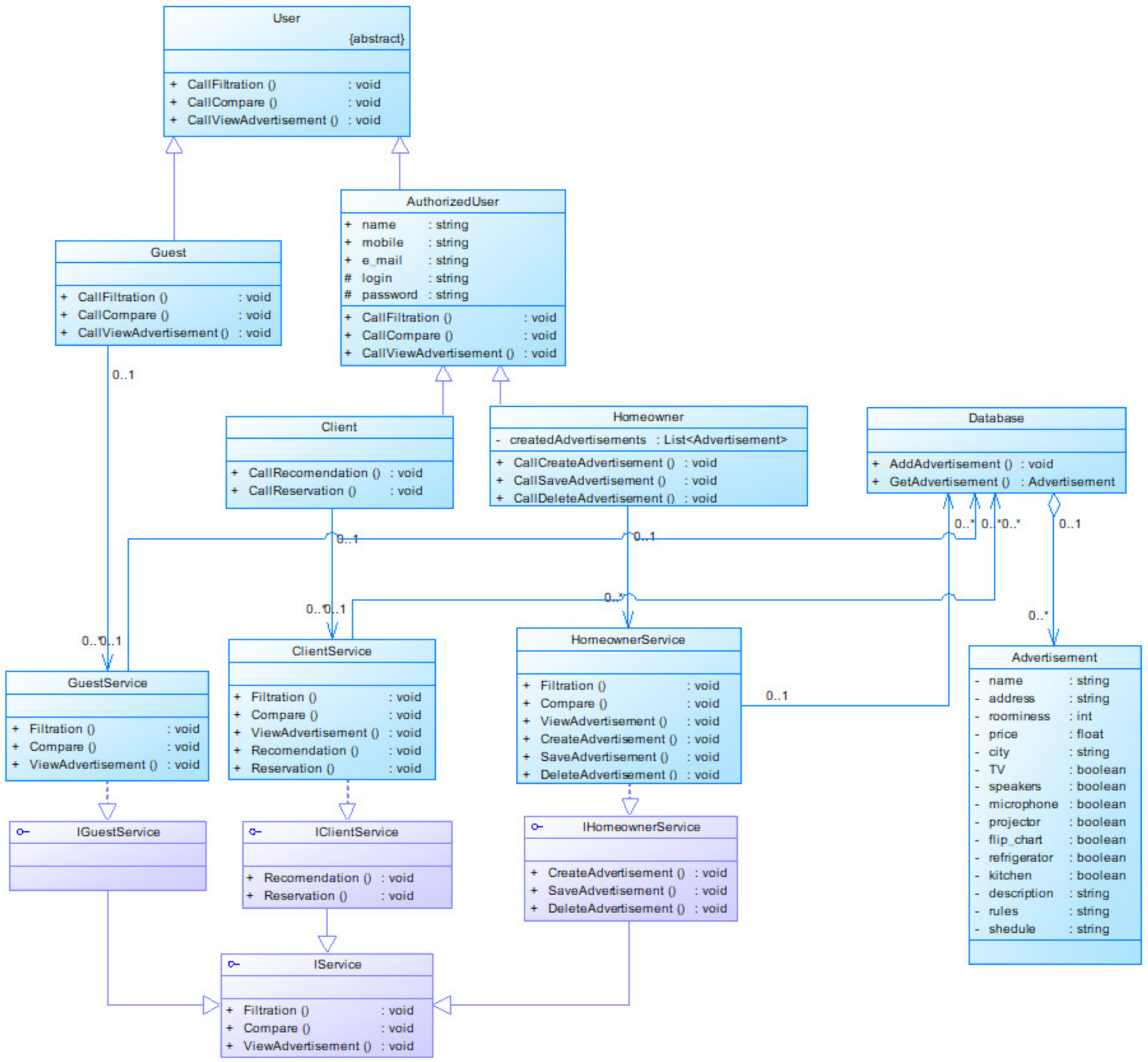
```
        public void UpdateDatabase()
        {
            db.SaveChanges();
        }

        public void DeleteAdvertisements(List<ModelAdvertisement> advertisements)
        {
            db.Advertisements.RemoveRange(advertisements);
            db.SaveChanges();
        }

        public void SaveAdvertisement(ModelAdvertisement advertisement)
        {
            db.Advertisements.Add(advertisement);
            db.SaveChanges();
        }

    }
```

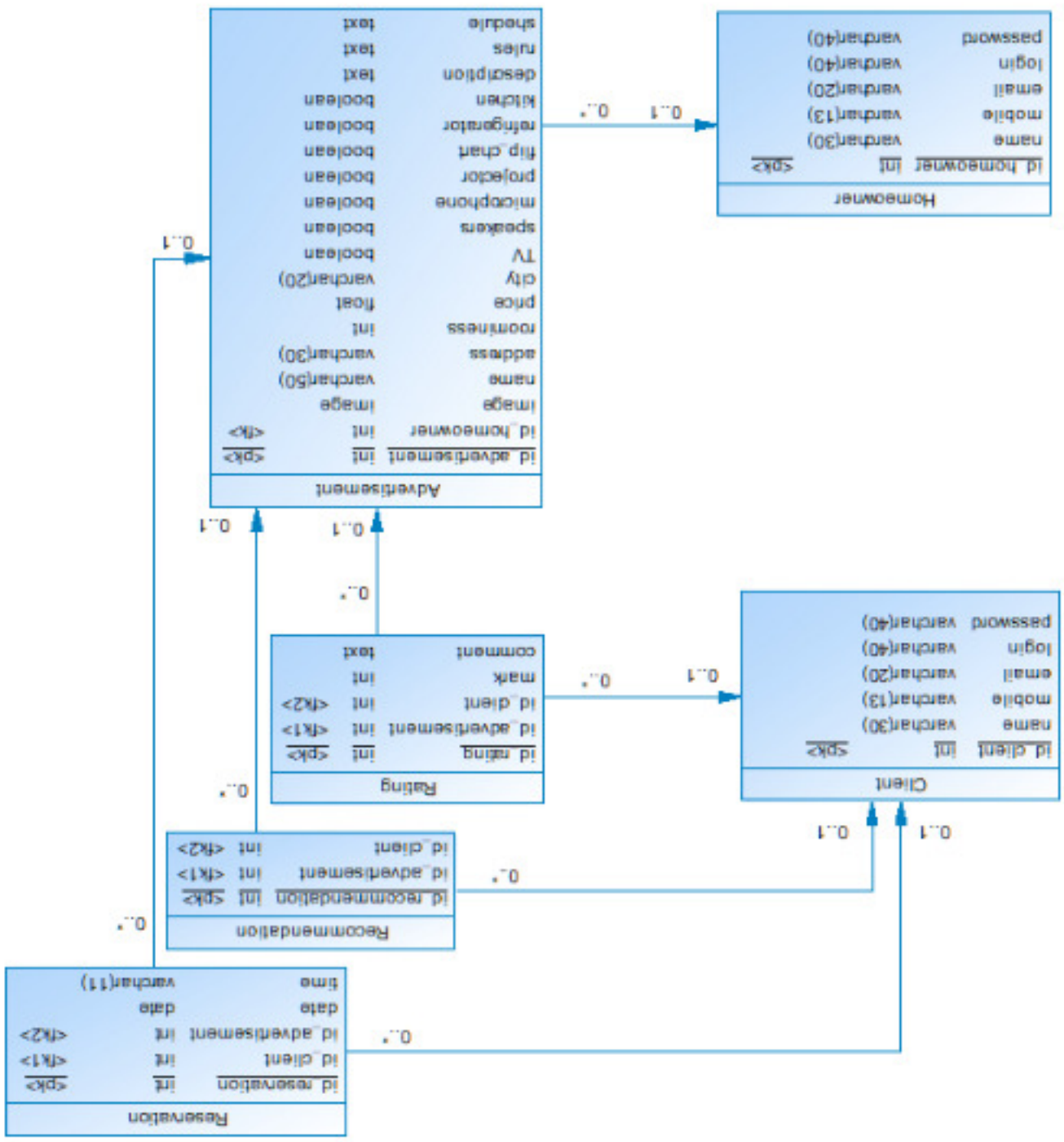
ДП ІС-5204.1181-с.ССК



					ДП ІС-5204.1181-с.ССК
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна класів програмного забезпечення
Розроб.		Брайко К.А.			Лит. / Маса / Масштаб
Перев.		Ковтунець О.В.			Аркуш 1 / Аркушів 1
Т. Кон.					Задача контентної фільтрації для організації конференцій
Н. Кон.		Халус О.А.			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Затв.		Ковтунець О.В.			

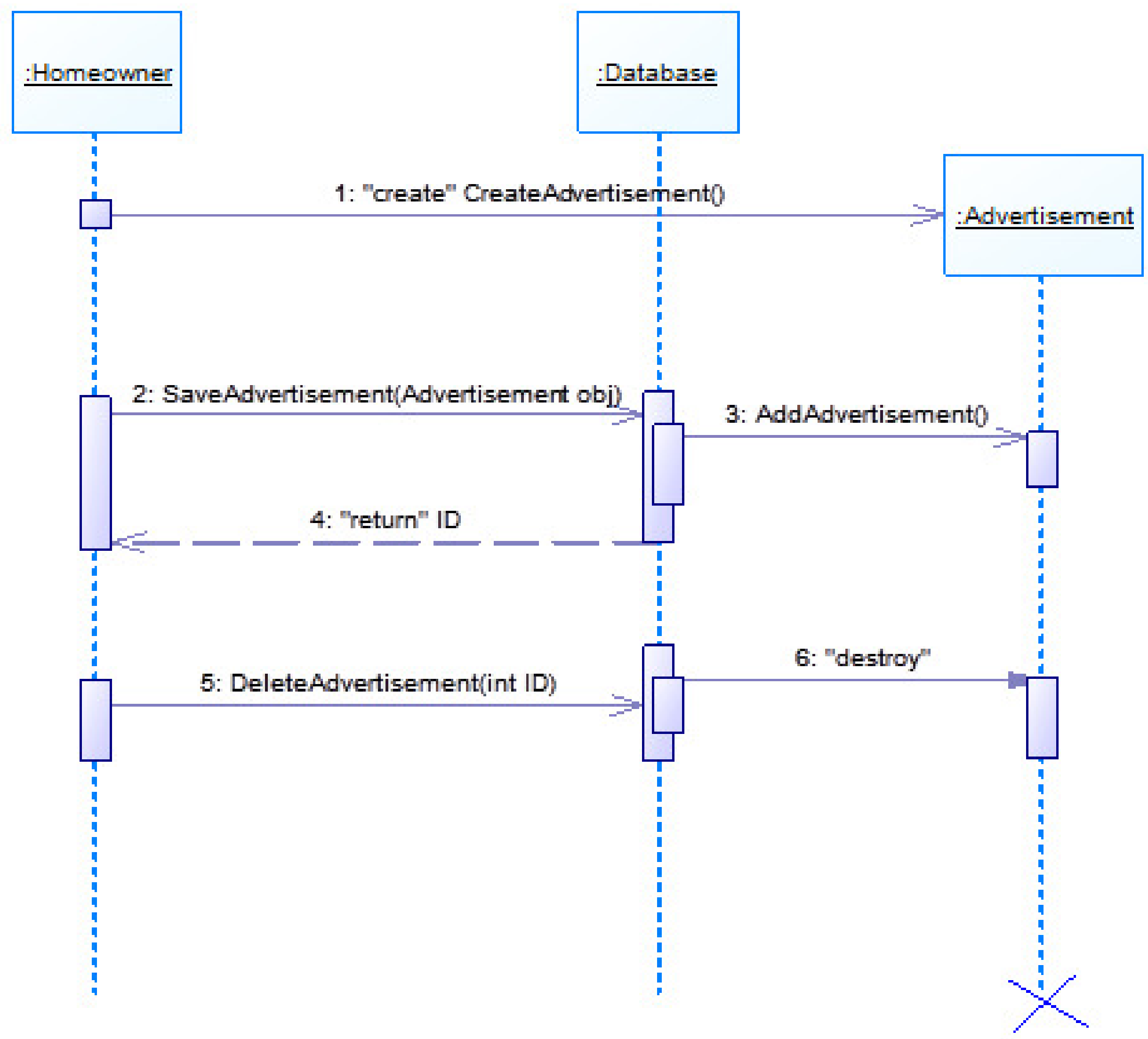
Зм.	Арк.	№ документа	Підпис	Дата							
Розробив		Бранько К.А.			Схема структура компонентів програмного забезпечення						
Перевірив		Ковтунець О.В.									
Т. контр.											
Н. контр.		Хапус О.А.			Задача контентної фільтрації для організації конференцій						
Затвердив		Ковтунець О.В.									
						Літера	Маса	Масштаб			
						Аркуш 1		Аркушів 1			

Зм.	Арк.	№ документа	Підпис	Дата	ДП ІС-5204.1181-с.СБД		
Розробив		Бранько К.А.			Схема бази даних	Літера	Маса	Масштаб
Перевірив		Ковтунець О.В.				Аркуш 1	Аркушів 1
Т. контр.					Задача контентної фільтрації для організації конференцій	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н. контр.		Харус О.А.					
Затвердив		Ковтунець О.В.					

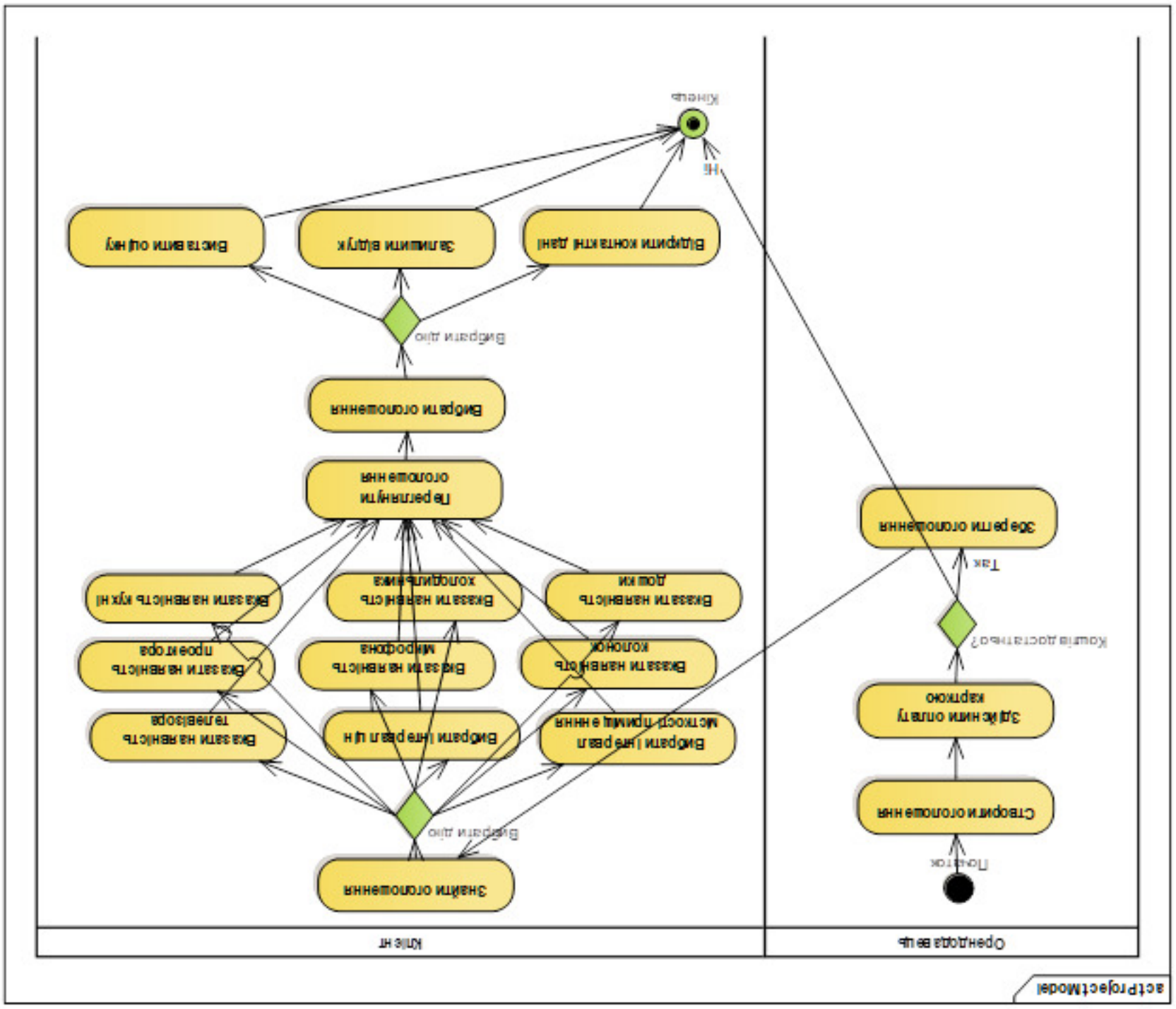


ДП ІС-5204.1181-с.СБД

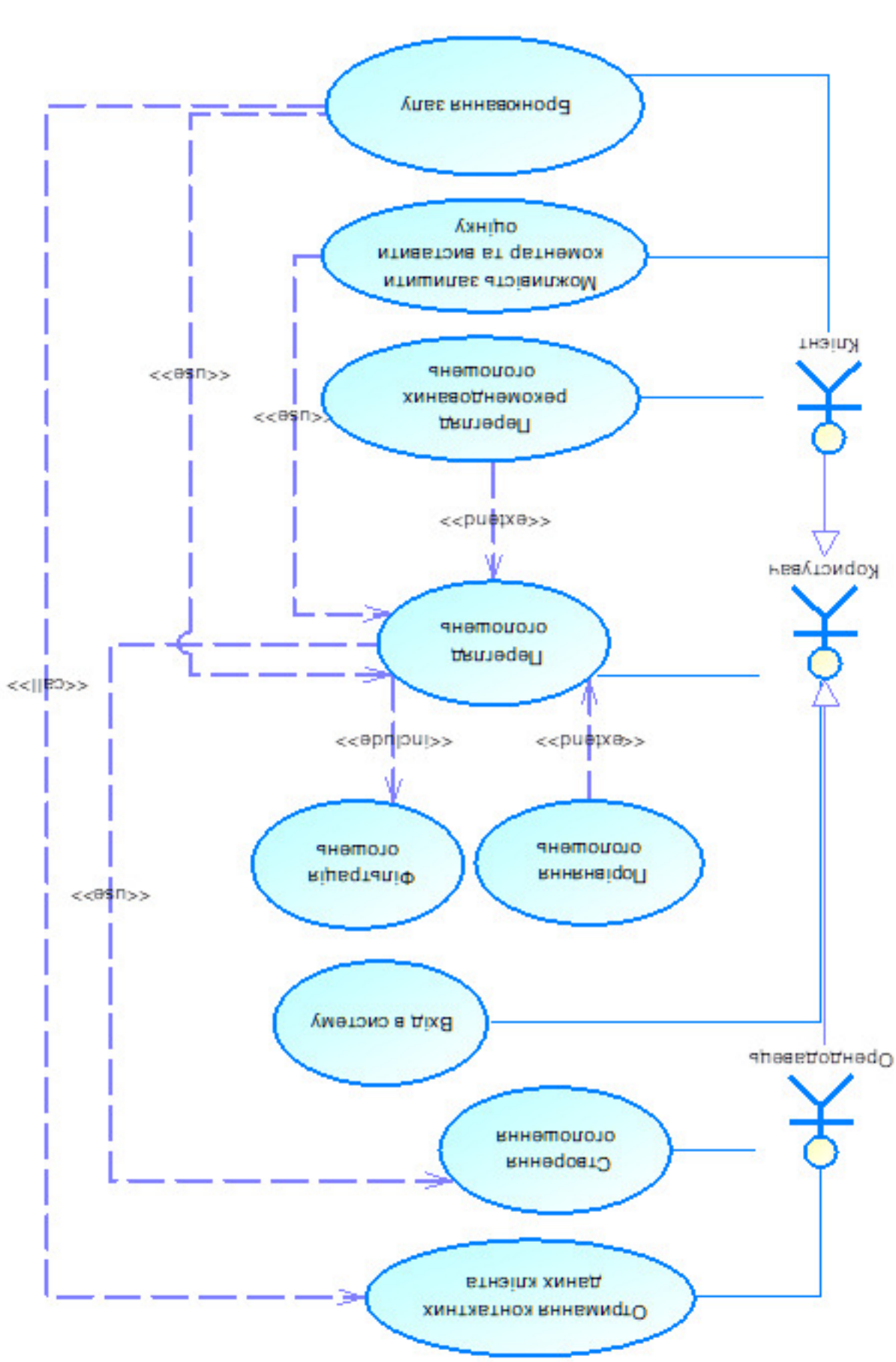
ДП ІС-5204.1181-с.ССП



Зм.	Арк.	№ документа	Підпис	Дата	ДП ІС-5204.1181-с.ССП
Розробив		Брайко К.А.			Схема структурна послідовності
Перевірив		Ковтунець О.В.			Літера / Маса / Масштаб; Аркуш 1 / Аркушів 1
Т. кон.					Задача контентної фільтрації для організації конференцій
Н. кон.		Халус О.А.			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Затвердив		Ковтунець О.В.			


actProjectModel
Клієнт
Організація
Старт
Створити оголошення
Задіснити оплату картою
Так
Перевірити оголошення
Зберегти оголошення
Вибрати дію
Знайти оголошення
Вибрати категорію
Вибрати категорію нерухомості
Вибрати назву товару
Вибрати назву продукта
Вибрати назву проектора
Вибрати назву ноутбука
Вибрати назву холодильника
Вибрати назву ліжка
Вибрати назву кухні
Вибрати дію
Відкрити контактні дані
Залишити відгук
Вистивити оцінку
Кінець

Зм.	Арк.	№ документа	Підпис	Дата	ДП ІС-5204.1181-с.ССВ				
Розробив		Бранько К.А.			Схема структура варіантів використань		Літера	Маса	Масштаб
Перевірив		Ковтунець О.В.					Аркуш 1	Аркушів 1	
Т. контр.					Задача контентної фільтрації для організації конференцій		КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52		
Н. контр.		Харус О.А.							
Затвердив		Ковтунець О.В.							



ДП ІС-5204.1181-с.ССВ